AFRL-IF-WP-TR-2003-1539

RAPID-PROTOTYPING OF APPLICATION SPECIFIC SIGNAL PROCESSORS (RASSP) EDUCATION AND FACILITATION

South Carolina Research Authority
5300 International Blvd.
N. Charleston, SC 29418

Advanced Technology Institute

DECEMBER 2000

Final Report for 13 June 1994 – 31 December 2000

Approved for public release; distribution is unlimited.

INFORMATION DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7334

NOTICE

USING GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA INCLUDED IN THIS DOCUMENT FOR ANY PURPOSE OTHER THAN GOVERNMENT PROCUREMENT DOES NOT IN ANY WAY OBLIGATE THE US GOVERNMENT. THE FACT THAT THE GOVERNMENT FORMULATED OR SUPPLIED THE DRAWINGS, SPECIFICATIONS, OR OTHER DATA DOES NOT LICENSE THE HOLDER OR ANY OTHER PERSON OR CORPORATION; OR CONVEY ANY RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY RELATE TO THEM.

THIS REPORT IS RELEASABLE TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). AT NTIS, IT WILL BE AVAILABLE TO THE GENERAL PUBLIC, INCLUDING FOREIGN NATIONS.

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

Maya Rubeiz

MAYA RUBEIZ
Project Engineer
Embedded Information Sys. Eng. Branch
Information Technology Division
Air Force Research Laboratory

Alfred J. Scarpelli

ALFRED J. SCARPELLI
Team Leader
Embedded Information Sys. Eng. Branch
Information Technology Division
Air Force Research Laboratory

James S. Williamson

JAMES S. WILLIAMSON, Chief
Embedded Information Sys. Eng. Branch
Information Technology Division
Air Force Research Laboratory

Do not return copies of this report unless contractual obligations or notice on a specific document requires its return.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD-MM-YY)* December 2000	**2. REPORT TYPE** Final	**3. DATES COVERED *(From - To)*** 06/13/1994 – 12/31/2000

4. TITLE AND SUBTITLE RAPID-PROTOTYPING OF APPLICATION SPECIFIC SIGNAL PROCESSORS (RASSP) EDUCATION AND FACILITATION	**5a. CONTRACT NUMBER** F33615-94-C-1457
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 63739E
6. AUTHOR(S)	**5d. PROJECT NUMBER** A268
	5e. TASK NUMBER 02
	5f. WORK UNIT NUMBER 11
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) South Carolina Research Authority 5300 International Blvd. N. Charleston, SC 29418 Advanced Technology Institute	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Information Directorate Air Force Research Laboratory Air Force Materiel Command Wright-Patterson AFB, OH 45433-7334 DARPA Tactical Technology Office 3701 Fairfax Drive Arlington, VA 22203-1714	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL/IFTA
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-IF-WP-TR-2003-1539

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

Report contains color. This report is also available on the companion CD-ROM when the AD'M number is ordered from DTIC (the AD'M additionally contains Appendix C, an HTML-based tutorial which is available only on the CD-ROM).

14. ABSTRACT

The Rapid-Prototyping of Application Specific Signal Processors (RASSP) program was a major DARPA/Tri-Service initiative to reinvent the process by which embedded digital signal processors were developed. The goal of the DARPA/Tri-service RASSP program was to dramatically improve the design process for complex digital systems, particularly embedded signal processors. A key objective was to reduce the total product development time by at least a factor of four while making similar improvements in product quality and life cycle cost. Also important was the ability to field state-of-the-art equipment at system build time and to rapidly upgrade the system throughout its life cycle. The RASSP Education and Facilitation (E&F) program was an unprecedented program set up to disseminate the information developed by 24 other RASSP programs to enable a paradigm shift in the way signal processors were designed. The RASSP E&F program is made up of four distinct functions: education, information server, interface, and transition. Major accomplishments include the establishment and maintenance of a webserver, the publication of the "RASSP Digest," development of educational and training materials, and technology transfer including executive seminars, workshops, RASSP Course Modules, and IEEE publication of two RASSP CD-ROMs. The second edition of the RASSP CD-ROM presents the essence of the knowledge from the entire RASSP program. The webserver has been relocated but is still operational.

15. SUBJECT TERMS

RASSP, RASSP E&F, RASSP Education and Facilitation, application specific signal processors, RASSP Digest, RASSP modules, VHDL Interactive Tutorial, VHDL – Electronic Systems Design Methodologies and Interactive Tutorial, Model Year Architecture

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	292	Maya Rubeiz **19b. TELEPHONE NUMBER** *(Include Area Code)* (937) 255-6653 x3593

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ADL	Arthur D. Little
ASEE	American Society of Engineering Educators
ATI	Advanced Technology Institute
CD-ROM	Compact Disk Read Only Memory
DAC	Design Automation Conference
DARPA	Defense Advanced Research Projects Agency
DSP	Digital Signal Processor
GT	Georgia Institute of Technology
GOMAC	Government Microcircuit Applications Conference
IEEE	Institute of Electrical and Electronics Engineers
EE	Electrical/Electronics
HTML	Hypertext Markup Language
HW/SW	Hardware/Software
NASA	National Aeronautics and Space Administration
PDF	Portable Document Format
RASSP	Rapid Prototyping of Application Specific Signal Processors
RASSP E&F	RASSP Education and Facilitation
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VIUF	VHDL International Users Forum
VLSI	Very Large Scale Integration
UC	The University of Cincinnati
UVA	University of Virginia

1 EXECUTIVE SUMMARY

The RASSP Education and Facilitation (E&F) program was the first Department of Defense program dedicated to extracting technology developed in a major research and development initiative and inserting it into industry and academia. As a result of this multi-year DARPA/Tri-Service effort, the technologies and methodologies developed by the Prime, Technology Base and Benchmarker contractors have been inserted into industry and academia. This method has allowed technology to be accessible for use without the usual gap of years between technology development and its widespread use. The success of RASSP E&F can be measured not only in terms of what was done during the program, but also in terms of the legacy provided by a book, CD-ROMs, course modules, a website and the students and faculty who have been exposed to RASSP technology.

The RASSP webserver has provided a means to disperse information worldwide to government, industry and academia. With over two million hits, the web page has proven to be a significant tool to reach an ever-expanding audience, regardless of location or time constraints. The *RASSP Digest* newsletter provided a professional journal that was keyed to the most significant audience, with thousands of copies delivered. The newsletter provided articles and news written by experts in engineering and education.

Over 3,000 EE degrees are awarded annually by 264 engineering schools in the United States. Over 70 of these schools now use RASSP modules as part of their teaching methodology. Over 800 copies of the first edition of the RASSP CD were purchased by practicing engineers and academia. To further enhance the penetration of the RASSP methodology into academia, complimentary copies of the second edition of the RASSP CD were sent to electrical engineering department heads of 107 universities and colleges. These statistics show the success of the program in its goal to insert the latest RASSP technology into educational programs and provide practicing engineers access to the RASSP technology.

2 INTRODUCTION

Rapid-Prototyping of Application Specific Signal Processors (RASSP) program was a major DARPA/Tri-Service initiative to reinvent the process by which embedded digital signal processors were developed. The goal of the DARPA/Tri-service RASSP program was to dramatically improve the design process for complex digital systems, particularly embedded signal processors. A key objective was to reduce the total product development time by at least a factor of four while making similar improvements in product quality and life cycle cost. Also important was the ability to field state-of-the-art equipment at system build time and to rapidly upgrade the system throughout its life cycle. RASSP met many of these goals through a combination of advanced design methodology emphasizing virtual prototyping, concurrent engineering, and design re-use; modular, scalable signal processor architectures; and a comprehensive supporting base of electronic design infrastructure, including automation tools, hardware and software libraries, enterprise integration capabilities, and standards. The program adopted an incremental refinement "model year" design methodology as a way of stressing the importance of continuous improvement, meeting short development schedules (3 to 12 months), and avoiding point design solutions. The model year methodology requires that systems be upgradable on an annual basis, with increasing function and performance. Many of the results of the RASSP program have been applied to other classes of electronic systems. Appendix A contains a list of RASSP program participants.

The RASSP program also pioneered two innovative concepts for managing a process-oriented program. First, development teams were benchmarked with semiannual "quizzes" --small design exercises that provided the feedback needed for continuous process improvement. The quizzes were based on design of a synthetic aperture radar image formation processor. Second, the program included an Educator/Facilitator contractor with explicit responsibility to ensure that RASSP design technology transitions effectively to the electronics community at large and continued to mature after completion of the RASSP program.

○ **“...In some specialties, engineers must update half of everything they know every couple of years…”** Ernest T. Smerdon

○ **“A paradigm change in design methodologies from I.C. implementation to system integration is mandated.”** James Freedman

The goal of the RASSP Education and Facilitation program was to develop a state-of-the-art education system to accelerate RASSP technology transfer to industry, government and academia. Over the past decade, electronic system capabilities have grown rapidly and manufacturing has made great strides in keeping up with this growth. However, design and verification techniques have fallen behind manufacturing. Furthermore, the gap between academia and real-world capabilities has widened significantly. Very few graduating engineers have the skills and knowledge needed by industry in the area of digital signal processor design. Educational techniques have not kept pace with technological advances. In addition, industry management does not have sufficient information to make technology pay-off decisions, especially when faced with the lag between current knowledge and current technology. These problems hinder technical growth and increase design time.

To overcome this problem and accomplish its goals, RASSP E&F developed and provided methodologies to enable change in the way embedded design was presented to the engineering community, how it was taught in universities, and how it was presented to industry decision-makers. Table 1 describes these steps used to accomplish the RASSP E&F's goal. In accomplishing it goal of developing a state-of-the-art education system to accelerate RASSP technology transfer to industry, RASSP E&F had to enable a significant paradigm shift in embedded system design education.

Table 1 - Steps to Accomplish the Goal of the RASSP Education and Facilitation Program

- Generate awareness and elicit user interest by providing a single point source of information about RASSP
- Develop, incorporate and disseminate an education system for university and continuing education that will facilitate the teaching of state-of-the-art system design techniques
- Educate senior executives in industry and government on the benefits of RASSP technology

Much of the success of the RASSP E&F program has been due to the diversity of the team. The team consisted of two levels of participants. Major participants provided much of the technical work and managerial support. Supporting participants provided additional specialized expertise on specific tasks. The RASSP E&F program was led by SCRA's Advanced Technology Institute (ATI) which provided program management and technical support.

Major participants included:

- The University of Virginia (UVA) which provided technical expertise in the area of VHSIC {Very High Speed Integrated Circuit} Hardware Description Language (VHDL), RASSP methodologies and instructional methodologies;
- Georgia Institute of Technology (GT) which provided technical expertise in the area of VHDL, RASSP methodologies and instructional methodologies;
- The University of Cincinnati (UC) which provided technical expertise in the area of VHDL and RASSP methodologies;
- Raytheon which provided expertise in the area of electronic manufacturing and technologies; and
- Arthur D. Little (ADL) which provided program management support.

Supporting participants included:

- Pennsylvania State University which provided assistance with the *VHDL: Electronic Systems Design Methodologies and Interactive Tutorial* CD-ROM;
- Mississippi State University which provided assistance with the *VHDL: Electronic Systems Design Methodologies and Interactive Tutorial* CD-ROM;
- Web Services, Incorporated which provided assistance with the *VHDL Interactive Tutorial* CD-ROM and the RASSP webserver;
- Enterprise Integration Technologies which provided assistance with the RASSP webserver; and
- Merkle and Mears which provided assistance in the area of current design methodologies.

3 ACCOMPLISHMENTS

Over 264 engineering schools in the U.S. grant electrical engineering (EE) degrees, and RASSP E&F interacted with over 100 of these. Approximately 38% of the colleges and universities in the U.S. who grant EE degrees have been impacted by this program. Over 70 schools have used modules developed by RASSP E&F. Major accomplishments include the establishment and maintenance of a webserver, the publication of the *RASSP Digest*, development of educational and training materials, and technology transfer. These accomplishments are discussed in the following sections.

3.1 RASSP Webserver

The RASSP Webserver was a key element, providing a central location source for information about the program. The website provides VHDL models and numerous RASSP Documents on-line as well as links to related DSP and VHDL sites. IEEE recognized the RASSP website as one of the "Top-3" in DSP. From the beginning of the program to 31 July 1999, a total of 2,493,745 requests were made from the website and 25,216.2 megabytes of data were transferred. During the course of the program, it was frequently accessed by engineers and students from many United States commercial and educational institutions as well as from many foreign countries. Tables 2 and 3 provide a list of educational and commercial institutions that frequently visited the RASSP website. Figure 1 shows the top twenty-seven foreign countries that visited the RASSP website.

Table 2- Selected List of Educational Visitors to the RASSP Website

Educational Visitors to the RASSP Website
Arizona State University
Auburn University
California Institute of Technology
California State University - Northridge
Case Western Reserve University
Clarkson University
Clemson University
Florida International University
Howard University
Massachusetts Institute of Technology
Michigan State University

Educational Visitors to the RASSP Website
North Carolina State University
Northwestern Polytechnic University
Oregon State University
Pennsylvania State University
Princeton University
Portland State University
Rice University
Rutgers University
San Jose State University
Stanford University
Texas A&M University - Kingsville
University of Arkansas - Fayetteville
University of California - Davis
University of California - Santa Barbara
University of Central Florida
University of Cincinnati
University of Illinois - Urbana Champaign
University of Massachusetts
University of Minnesota
University of Missouri - Rolla
University of South Carolina
University of Texas at Austin
University of Virginia
Virginia Tech
Washington University

Table 3 - Selected List of Commercial Visitors to the RASSP Website

Commercial Visitors to the RASSP Website
Alexa Internet
America Online, Inc.
Banco Santander
Cadence Design Systems
Digital Equipment Corporation
EXCALIBUR Group, A Time Warner Company
Excite, Inc.
Harris Corporation
Home Network
Honeywell, Inc.
Hongkong Telecom IMS
IBM Corporation
Imagelock, Inc.

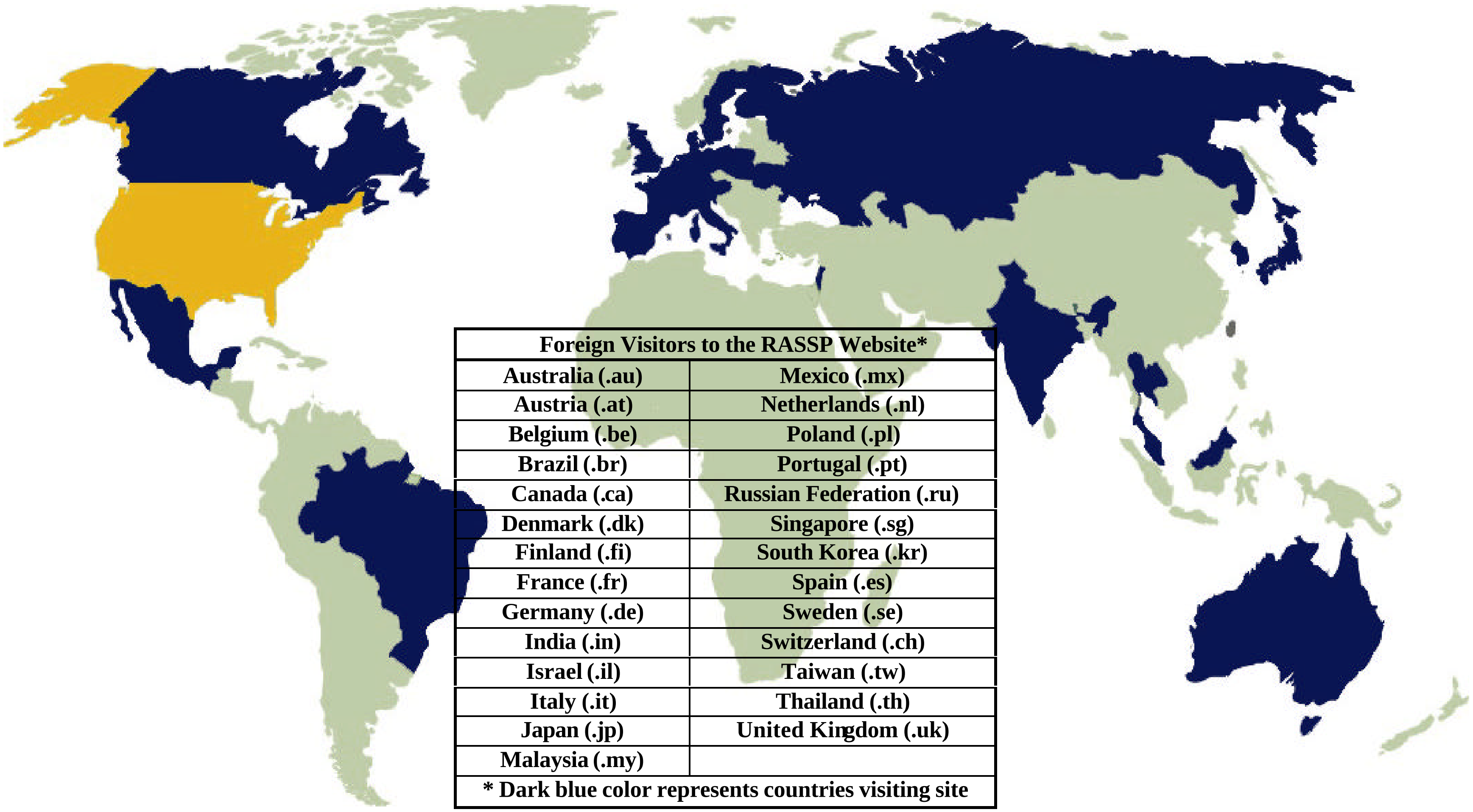
Commercial Visitors to the RASSP Website
Intel Corporation
JavaNet
Mentor Graphics Corporation
NETCOM On-Line Communication Services, Inc.
Nokia Head Office
Philips Electronics B.V.
Real Time Technologies, Inc.
Rockwell International Corporation
S3, Inc.
SAIC
Sara Lee Hosiery
Schlumberger Ltd.
Science & Applied Technology
Scudder, Stevens & Clark
Seagate Technology, Inc.
SGS-THOMSON Microelectronics
Shasta Networks
Siemens Business Communications Systems, Inc.
Siemens Research and Technology Laboratories
Sierra Imaging
Silicon Automation Systems
Silicon Dynamics
Silicon Logic Engineering
Silicon Valley Research, Inc.
Silicon Value
SiliconWave, Inc.
Simoco Telecommunications Ltd.
SISA
Smith International
Snow Hill Network Services
Software & Technologies, Inc.
Solidum Systems Corp.
Sony Corporation of America
South Bend Tribune
South Coast Computing Services, Inc.
Southern California Edison
Sovam Teleport
Spacebridge Communications Corporation
Spectrum Signal Processing, Inc.
Sterling Commerce
Stone Age Imaging, Inc.
Stroock & Stroock & Lavan
Sundstrand Corporation
Superonline
Surf South

Commercial Visitors to the RASSP Website
Sybron Dental Specialties
Synopsys, Inc.
Tec-Masters, Inc.
Texas Instruments
The Internet Access Company
The Silicon Group, Inc.
Thomson & Thomson
VORT Corporation
Webco International
Xilinx, Inc.

Figure 2, which follows, shows that the RASSP webserver site has been a very active site. The peak usage was in May 1997 with over 130,000 accesses. Figure 3 shows the number of unique hosts that accessed the webserver on a per month basis. Figure 4 shows the data transferred on a monthly basis, and Figure 5 shows the data transferred per unique host per month. The peak was in July 1998, with a transfer of over 1.5 megabytes per host, and continues at over 200,000 bytes per host in July 1999. This graph is important because it shows that website users are not just browsing, but are downloading the files from the webserver.

Activity on the website increased yearly as the RASSP program produced material that was posted on the website and published in the *RASSP Digest* newsletter. The RASSP program was very active in 1997 when the peak occurred, and accesses remain strong even after the prime and technology base efforts ended. See Appendix A for a list of the program participants for these efforts. The website has been available since the beginning of the program; however, major disruptions did occur due to hurricanes, telecommunications failure or computer failure. The last major update to the webserver was made in July 2000 with much of the RASSP material from the second edition of the CD-ROM posted. The RASSP webserver will continue to operate for at least six months after the RASSP E&F program ends. The URL for the RASSP website is http://rassp.scra.org.

Figure 1 - Selected List of Foreign Country Visitors to the RASSP Website



Foreign Visitors to the RASSP Website*	
Australia (.au)	Mexico (.mx)
Austria (.at)	Netherlands (.nl)
Belgium (.be)	Poland (.pl)
Brazil (.br)	Portugal (.pt)
Canada (.ca)	Russian Federation (.ru)
Denmark (.dk)	Singapore (.sg)
Finland (.fi)	South Korea (.kr)
France (.fr)	Spain (.es)
Germany (.de)	Sweden (.se)
India (.in)	Switzerland (.ch)
Israel (.il)	Taiwan (.tw)
Italy (.it)	Thailand (.th)
Japan (.jp)	United Kingdom (.uk)
Malaysia (.my)	
* Dark blue color represents countries visiting site	

Figure 2 - RASSP Webserver Accesses per Month

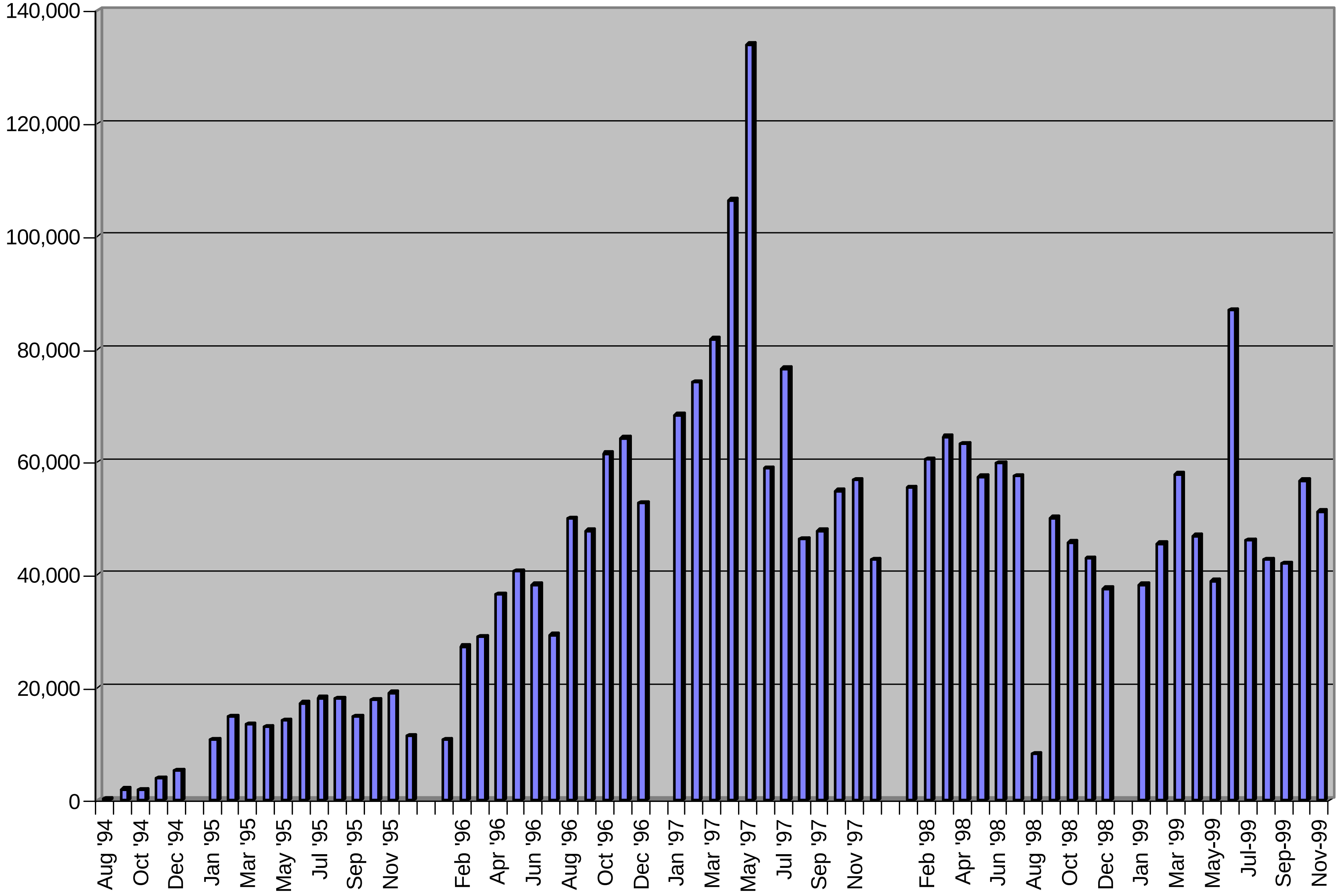


Figure 3 - RASSP Webserver Unique Hosts per Month

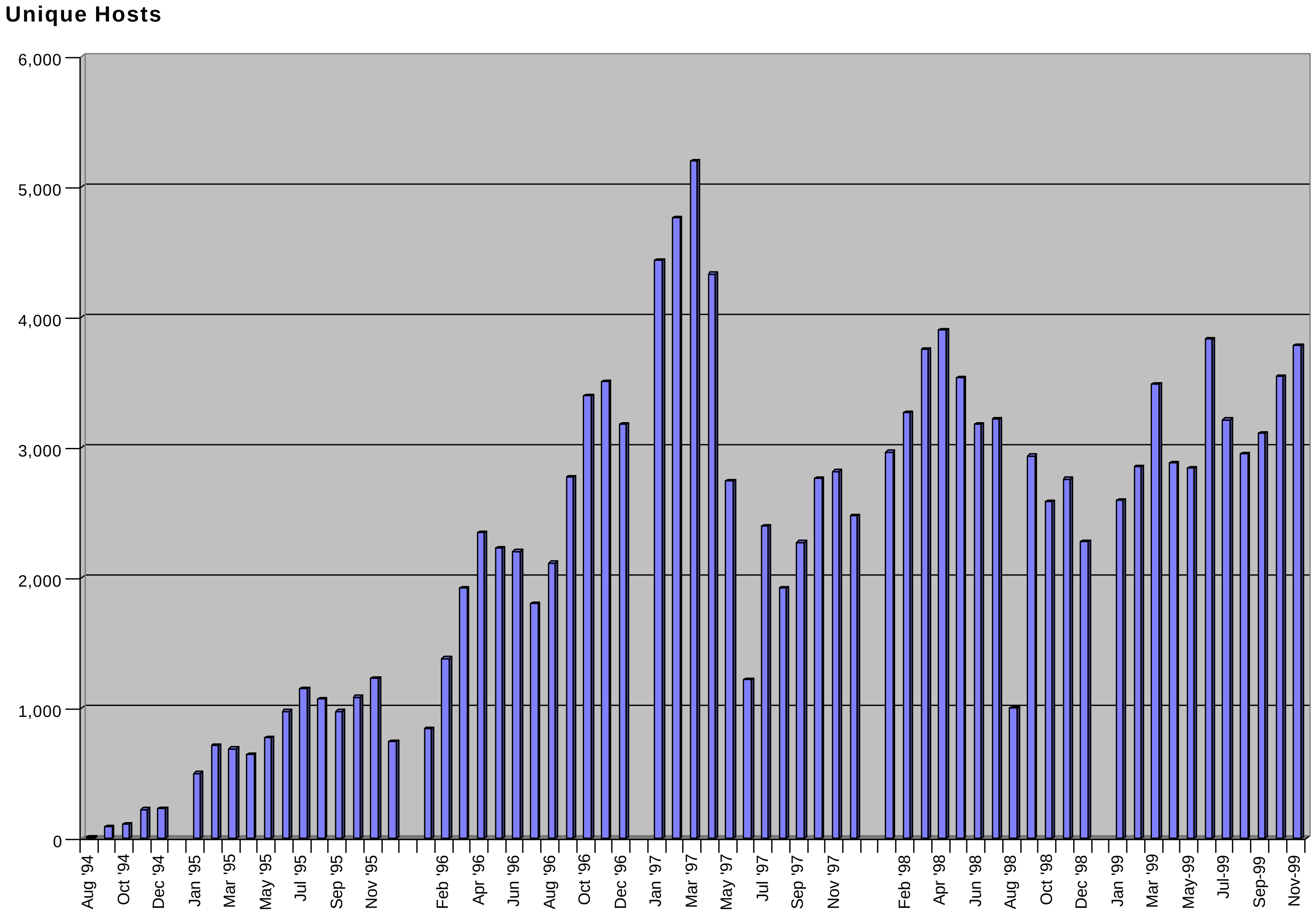


Figure 4 - Megabytes of Data Transferred per Month

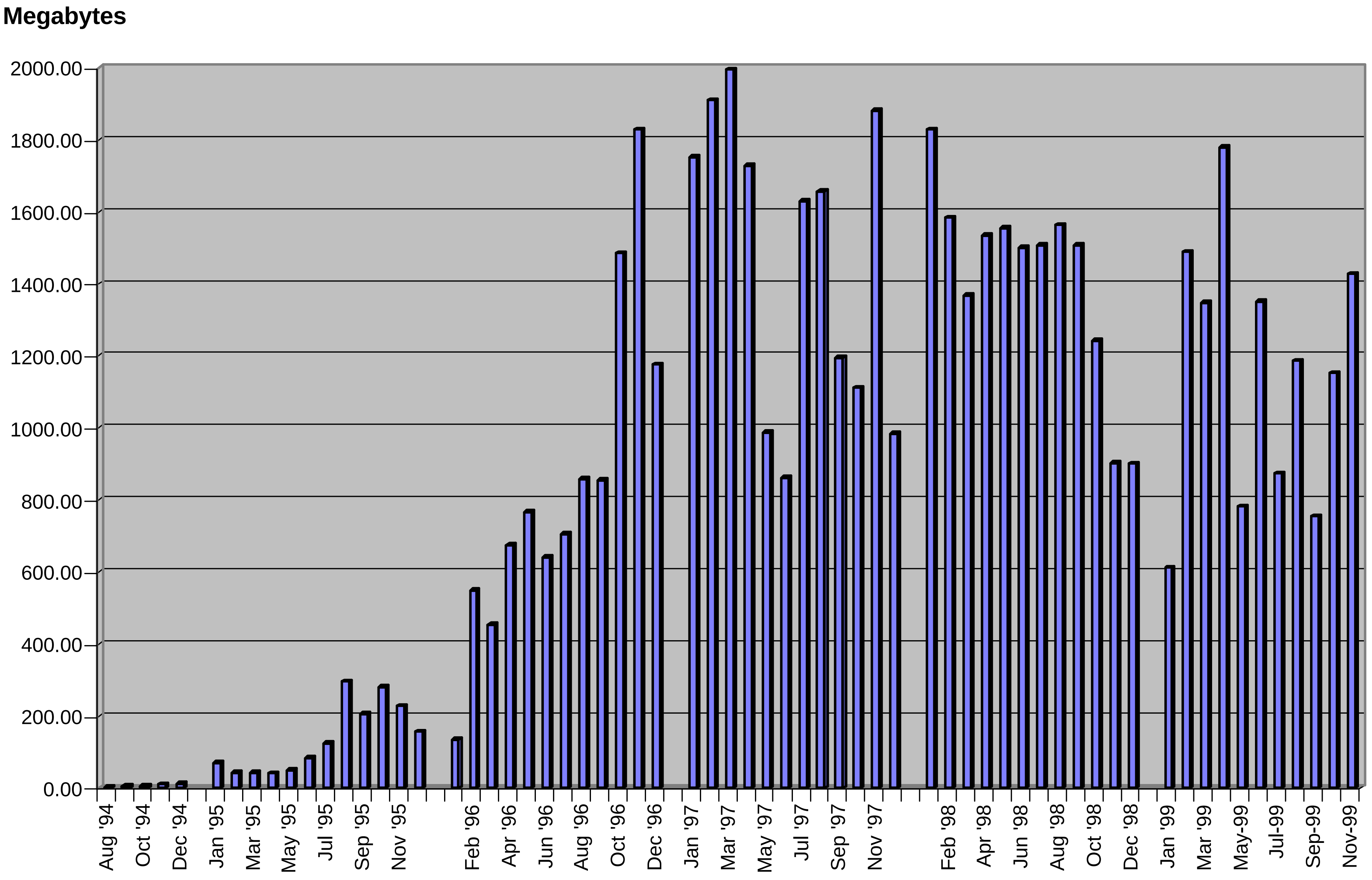
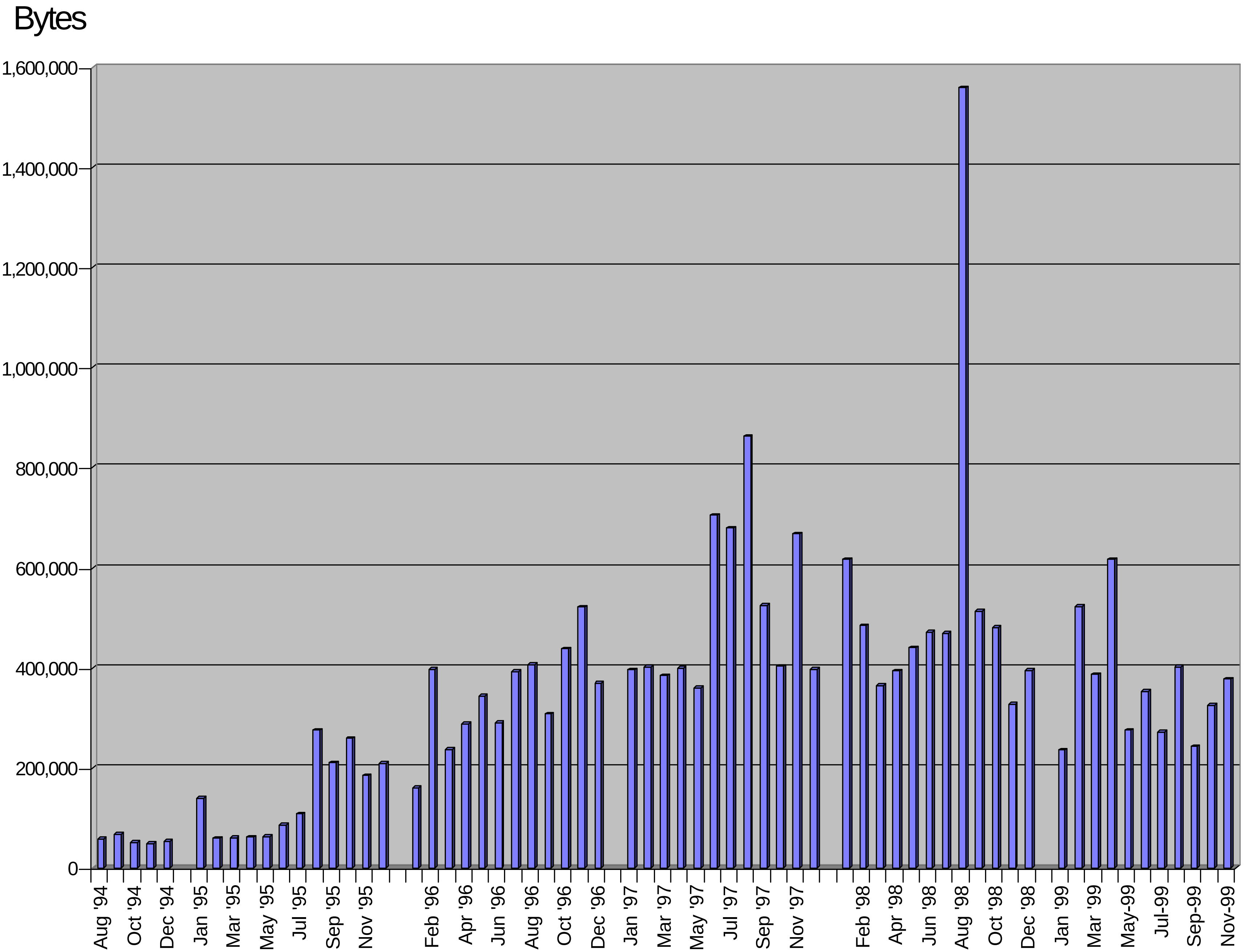


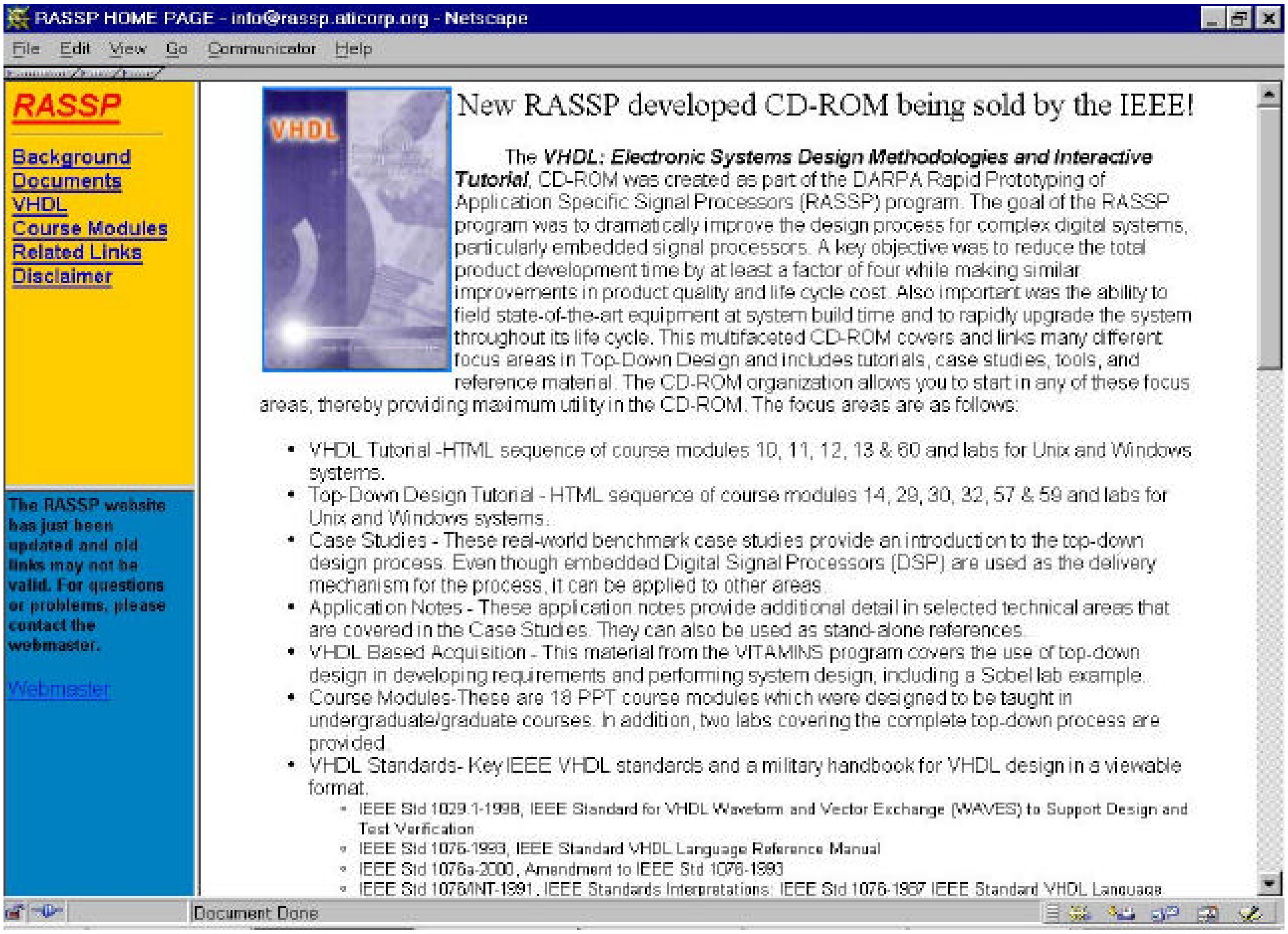
Figure 5 - RASSP Webserver Bytes Transferred per Unique Host per Month



The RASSP webserver underwent over six major modifications in layout over the course of the program. These changes were due to improvements in web technology, as well as changes in the data available. Another cause for updates was the increase in web expertise and expectations.

Figure 6 shows the RASSP Webserver homepage. Major items of interest are listed in the right frames, and the menus for traversing the site are in the left frames. Figure 7 shows part of the RASSP Document main page, and Figure 8 shows part of the *RASSP Digest* main page. The steps that a user would follow to obtain information in the VHDL area are shown in Figure 9 and elaborated in Figures 10, 11 and 12.

Figure 6 - RASSP Webserver Home Page



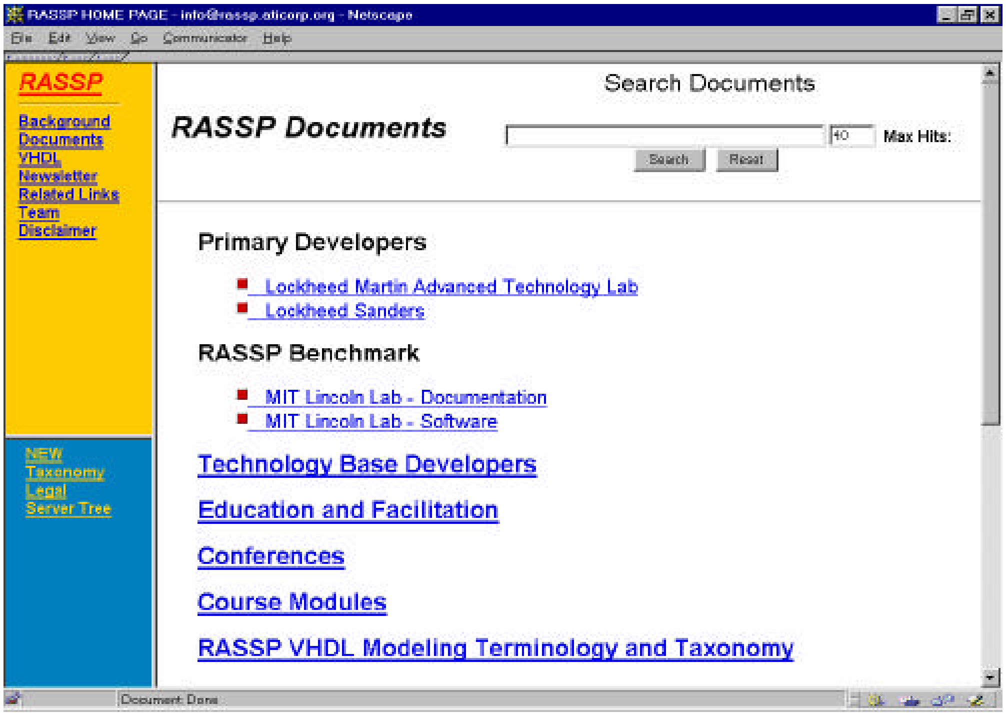
The RASSP website has just been updated and old links may not be valid. For questions or problems, please contact the webmaster.

New RASSP developed CD-ROM being sold by the IEEE!

The ***VHDL: Electronic Systems Design Methodologies and Interactive Tutorial***, CD-ROM was created as part of the DARPA Rapid Prototyping of Application Specific Signal Processors (RASSP) program. The goal of the RASSP program was to dramatically improve the design process for complex digital systems, particularly embedded signal processors. A key objective was to reduce the total product development time by at least a factor of four while making similar improvements in product quality and life cycle cost. Also important was the ability to field state-of-the-art equipment at system build time and to rapidly upgrade the system throughout its life cycle. This multifaceted CD-ROM covers and links many different focus areas in Top-Down Design and includes tutorials, case studies, tools, and reference material. The CD-ROM organization allows you to start in any of these focus areas, thereby providing maximum utility in the CD-ROM. The focus areas are as follows:

- VHDL Tutorial -HTML sequence of course modules 10, 11, 12, 13 & 60 and labs for Unix and Windows systems.
- Top-Down Design Tutorial - HTML sequence of course modules 14, 29, 30, 32, 57 & 59 and labs for Unix and Windows systems.
- Case Studies - These real-world benchmark case studies provide an introduction to the top-down design process. Even though embedded Digital Signal Processors (DSP) are used as the delivery mechanism for the process, it can be applied to other areas.
- Application Notes - These application notes provide additional detail in selected technical areas that are covered in the Case Studies. They can also be used as stand-alone references.
- VHDL Based Acquisition - This material from the VITAMINS program covers the use of top-down design in developing requirements and performing system design, including a Sobel lab example.
- Course Modules-These are 18 PPT course modules which were designed to be taught in undergraduate/graduate courses. In addition, two labs covering the complete top-down process are provided.
- VHDL Standards- Key IEEE VHDL standards and a military handbook for VHDL design in a viewable format.
  - IEEE Std 1029.1-1998, IEEE Standard for VHDL Waveform and Vector Exchange (WAVES) to Support Design and Test Verification
  - IEEE Std 1076-1993, IEEE Standard VHDL Language Reference Manual
  - IEEE Std 1076a-2000, Amendment to IEEE Std 1076-1993
  - IEEE Std 1076/INT-1991, IEEE Standards Interpretations: IEEE Std 1076-1987 IEEE Standard VHDL Language

Figure 7 - RASSP Documents Page


RASSP HOME PAGE - info@rassp.aticorp.org - Netscape
File Edit View Go Communicator Help
Document: Done

RASSP

Background
Documents
VHDL
Newsletter
Related Links
Team
Disclaimer

NEW
Taxonomy
Legal
Server Tree

Search Documents

RASSP Documents

Max Hits: 40

Search Reset

Primary Developers

- Lockheed Martin Advanced Technology Lab
- Lockheed Sanders

RASSP Benchmark

- MIT Lincoln Lab - Documentation
- MIT Lincoln Lab - Software

Technology Base Developers

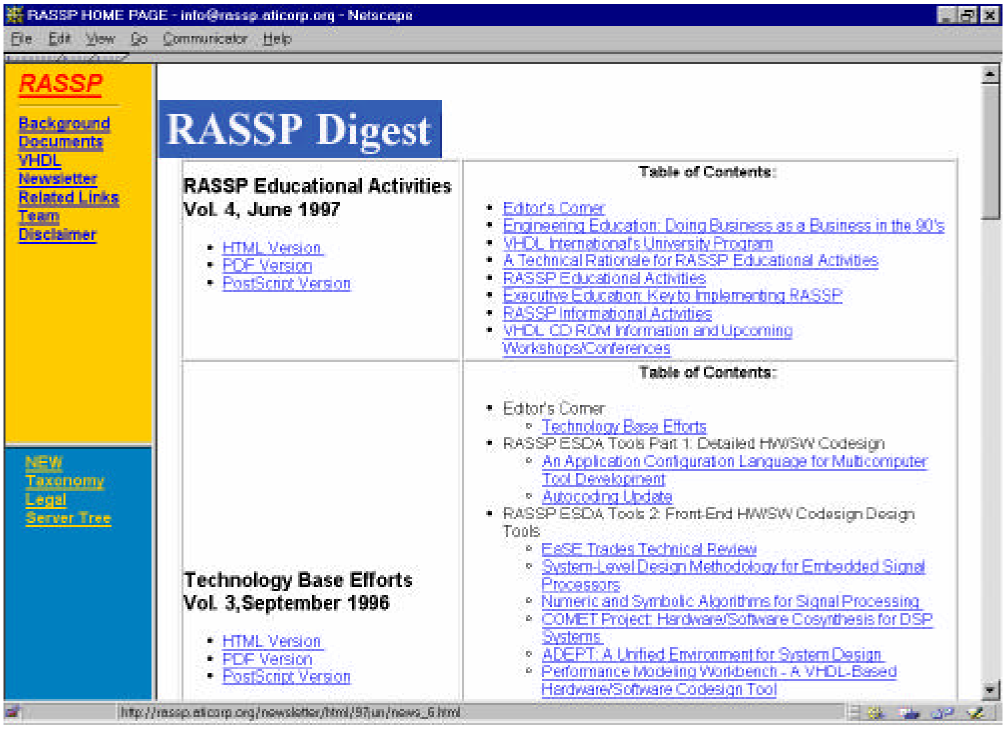
Education and Facilitation

Conferences

Course Modules

RASSP VHDL Modeling Terminology and Taxonomy

Figure 8 - *RASSP Digest* Page


RASSP HOME PAGE - info@rassp.aticorp.org - Netscape
File Edit View Go Communicator Help
http://rassp.aticorp.org/newsletter/html/97jun/news_6.html

RASSP

Background
Documents
VHDL
Newsletter
Related Links
Team
Disclaimer

NEW
Taxonomy
Legal
Server Tree

RASSP Digest

RASSP Educational Activities
Vol. 4, June 1997

- HTML Version
- PDF Version
- PostScript Version

Table of Contents:

- Editor's Corner
- Engineering Education: Doing Business as a Business in the 90's
- VHDL International's University Program
- A Technical Rationale for RASSP Educational Activities
- RASSP Educational Activities
- Executive Education: Key to Implementing RASSP
- RASSP Informational Activities
- VHDL CD ROM Information and Upcoming Workshops/Conferences

Technology Base Efforts
Vol. 3,September 1996

- HTML Version
- PDF Version
- PostScript Version

Table of Contents:

- Editor's Corner
  - Technology Base Efforts
- RASSP ESDA Tools Part 1: Detailed HW/SW Codesign
  - An Application Configuration Language for Multicomputer Tool Development
  - Autocoding Update
- RASSP ESDA Tools 2: Front-End HW/SW Codesign Design Tools
  - EaSE Trades Technical Review
  - System-Level Design Methodology for Embedded Signal Processors
  - Numeric and Symbolic Algorithms for Signal Processing
  - COMET Project: Hardware/Software Cosynthesis for DSP Systems
  - ADEPT: A Unified Environment for System Design
  - Performance Modeling Workbench - A VHDL-Based Hardware/Software Codesign Tool

Figure 9 – Example of Path to VHDL Memory Models

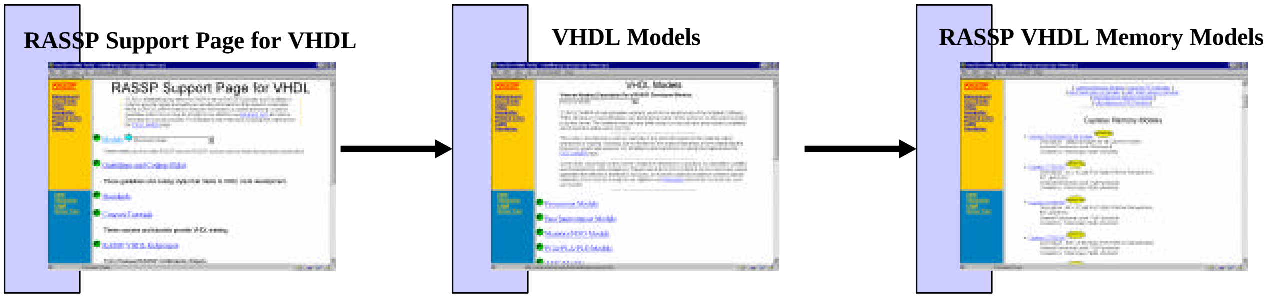
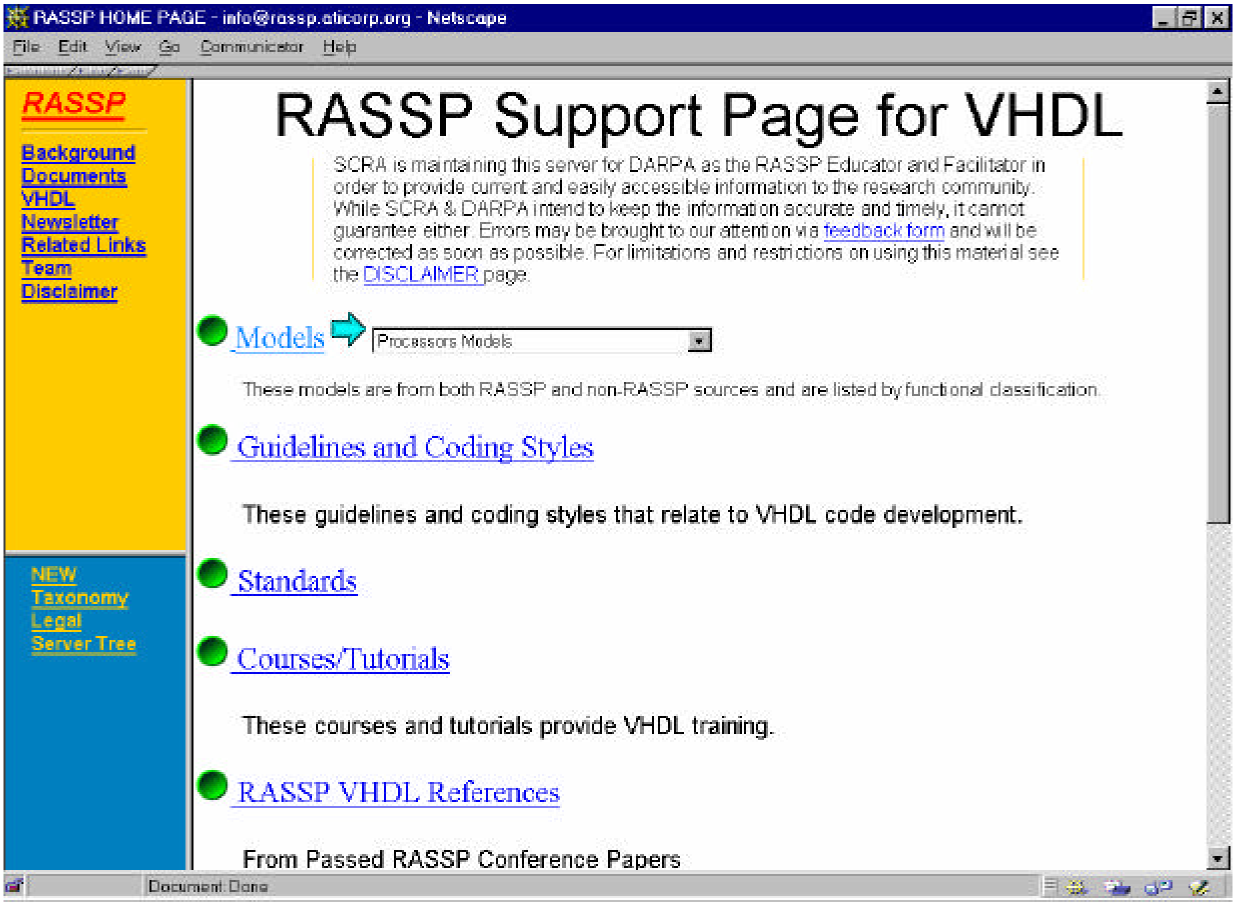


Figure 10 - RASSP VHDL Support Page



RASSP Support Page for VHDL

SCRA is maintaining this server for DARPA as the RASSP Educator and Facilitator in order to provide current and easily accessible information to the research community. While SCRA & DARPA intend to keep the information accurate and timely, it cannot guarantee either. Errors may be brought to our attention via feedback form and will be corrected as soon as possible. For limitations and restrictions on using this material see the DISCLAIMER page.

● Models Processors Models

These models are from both RASSP and non-RASSP sources and are listed by functional classification.

● Guidelines and Coding Styles

These guidelines and coding styles that relate to VHDL code development.

● Standards

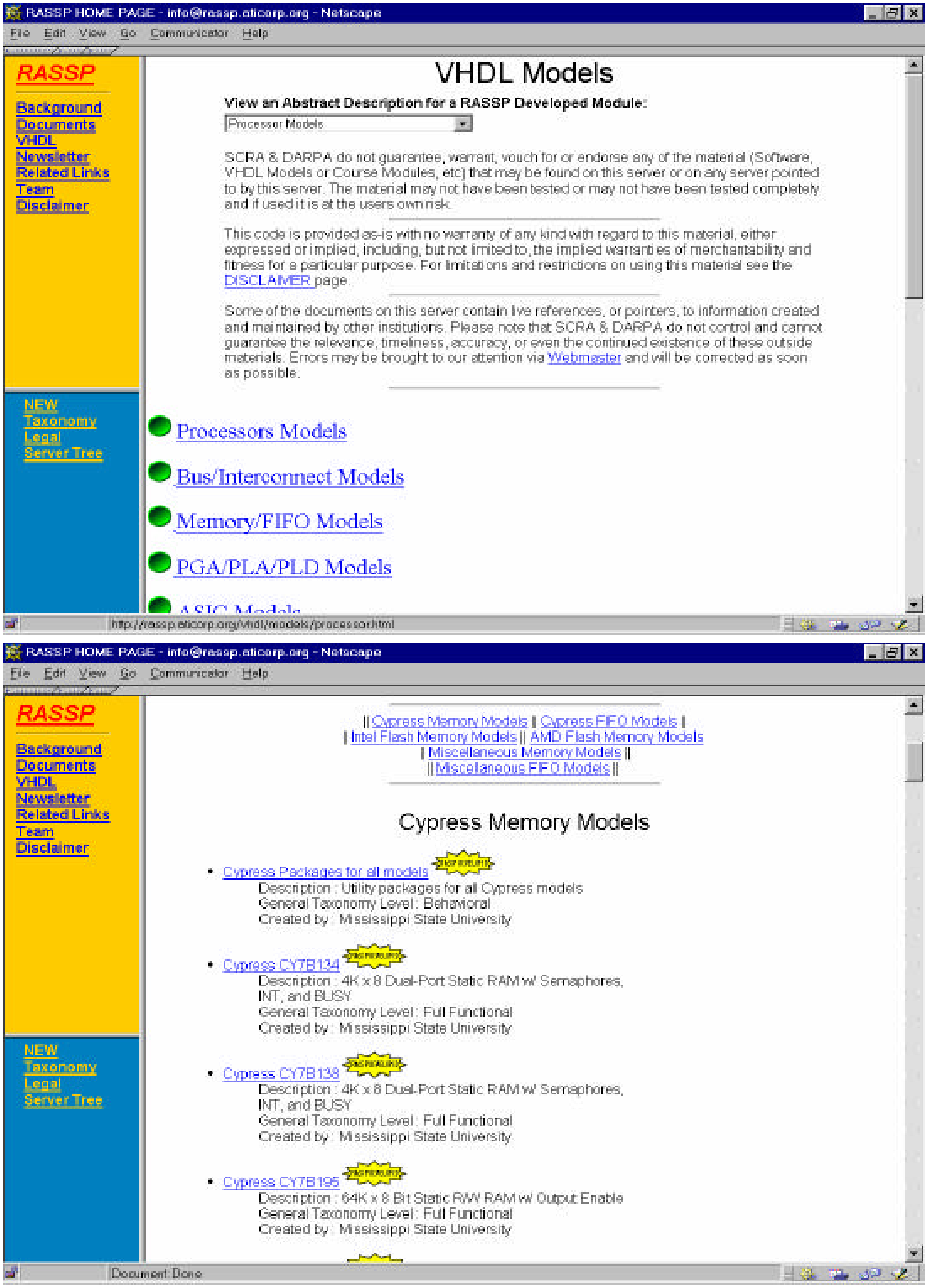
● Courses/Tutorials

These courses and tutorials provide VHDL training.

● RASSP VHDL References

From Passed RASSP Conference Papers

Figure 11 - RASSP VHDL Model Page



RASSP HOME PAGE - info@rassp.aticorp.org - Netscape
File Edit View Go Communicator Help
RASSP
Background
Documents
VHDL
Newsletter
Related Links
Team
Disclaimer
NEW
Taxonomy
Legal
Server Tree
Processor Models
http://rassp.aticorp.org/vhdl/models/processor.html

VHDL Models

View an Abstract Description for a RASSP Developed Module:

SCRA & DARPA do not guarantee, warrant, vouch for or endorse any of the material (Software, VHDL Models or Course Modules, etc) that may be found on this server or on any server pointed to by this server. The material may not have been tested or may not have been tested completely and if used it is at the users own risk.

This code is provided as-is with no warranty of any kind with regard to this material, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. For limitations and restrictions on using this material see the DISCLAIMER page.

Some of the documents on this server contain live references, or pointers, to information created and maintained by other institutions. Please note that SCRA & DARPA do not control and cannot guarantee the relevance, timeliness, accuracy, or even the continued existence of these outside materials. Errors may be brought to our attention via Webmaster and will be corrected as soon as possible.

- Processors Models
- Bus/Interconnect Models
- Memory/FIFO Models
- PGA/PLA/PLD Models
- ASIC Models

|| Cypress Memory Models | Cypress FIFO Models |
| Intel Flash Memory Models || AMD Flash Memory Models
| Miscellaneous Memory Models ||
|| Miscellaneous FIFO Models ||

Cypress Memory Models

- Cypress Packages for all models
  Description : Utility packages for all Cypress models
  General Taxonomy Level : Behavioral
  Created by : Mississippi State University
- Cypress CY7B134
  Description : 4K x 8 Dual-Port Static RAM w/ Semaphores, INT, and BUSY
  General Taxonomy Level : Full Functional
  Created by : Mississippi State University
- Cypress CY7B138
  Description : 4K x 8 Dual-Port Static RAM w/ Semaphores, INT, and BUSY
  General Taxonomy Level : Full Functional
  Created by : Mississippi State University
- Cypress CY7B195
  Description : 64K x 8 Bit Static R/W RAM w/ Output Enable
  General Taxonomy Level : Full Functional
  Created by : Mississippi State University

3.2 RASSP Digest

During the course of the program, several issues of the RASSP Digest were produced, providing a professional journalistic format for the promulgation of the RASSP concepts. Over 10,000 hard copies of the Digest have been distributed, with over 1,300 individuals on the hard copy mailing list. An additional 500 individuals received e-mail notification that the issue was posted on the RASSP website. Most of the individuals receiving either a hard copy or e-mail notification were industry or government engineers actively engaged in management or design of electronic systems, or were professors teaching VHDL or DSP. The digest featured articles written by experts in engineering, education and management. These articles focused on the primary development activities, the benchmarking activities, and the technology base developments, and reported conference events. On the webserver, links were available to HTML versions of each article as well as to a PDF version of the complete issue. Table 4 lists the table of contents of all issues of the digest. See external Appendix E for printed copies of the *RASSP Digest.*

Table 4 -Table of Contents for all *RASSP Digest*

Issue	Table of Contents
RASSP Educational Activities **Vol. 4, June 1997**	• Editor's Corner • Engineering Education: Doing Business as a Business in the 90's • VHDL International's University Program • A Technical Rationale for RASSP Educational Activities • RASSP Educational Activities • Executive Education: Key to Implementing RASSP • RASSP Informational Activities • VHDL CD-ROM Information and Upcoming Workshops/Conferences
Technology Base Efforts **Vol. 3, September 1996**	• Editor's Corner o Technology Base Efforts • RASSP ESDA Tools Part 1: Detailed HW/SW Codesign o An Application in Configuration Language for Multicomputer Tool Development o Autocoding Update • RASSP ESDA Tools 2: Front-End HW/SW Codesign Design Tools o EaSE Trades Technical Review o System-Level Design Methodology for Embedded Signal Processors o Numeric and Symbolic Algorithms for Signal Processing o COMET Project: Hardware/Software Cosynthesis for DSP Systems o ADEPT: A Unified Environment for System Design o Performance Modeling Workbench - A VHDL-Based Hardware/Software Codesign Tool

Issue	Table of Contents
	o ANSI C to Behavioral VHDL Translator, Ada to Behavioral VHDL Translator o MAT2DSP - A MATLAB Tool for the Automatic Evaluation of the Implementation Requirements of Signal Processing Algorithms o Timing Insensitive Binary-to-Binary Translation (TIBBIT) o Design Tools and Architectures for Dedicated DSP Processors • RASSP Model Libraries: VHDL o VHDL Hybrid Models o Automated Generation of VHDL Processor Models for Simulation and Synthesis o Mississippi State Develops On-Line FPGA VHDL Model Generator
The Road to Enterprise Integration **Vol. 3, 1st Qtr. 1996**	• Editor's Corner o The Road to Enterprise Integration • Prime Development o Integrated Process Control and Data Management in RASSP Enterprise Systems o Enterprise Integration • Technology Base o The National Industrial Information Infrastructure Protocols Project (NIIIP) o Concurrent Engineering Wheels o Agility Through Information Sharing: Results Achieved in a Production Environment
Model Year Architecture **Vol. 2, 4th Qtr. 1995**	• Editor's Corner o RASSP Model Year Architectures • Prime Development o The Standard Virtual Interface - An Interoperability Approach • Technology Base o A RASSP Approach to HW/SW Codesign o A Hierarchical and Integrated Built-in Self-Test Methodology o An Overview of Automated Processor Specification and Task Allocation Techniques for Embedded Computer Systems
RASSP at 24 Months **Vol. 2, 3rd Qtr. 1995**	• RASSP at 24 Months • The Second Annual RASSP Conference, a Mid-Program Review • Second Annual RASSP Conference, Synopsis of Session 2, "Introduction to RASSP and 2nd Year Overview" • Sanders RASSP Program Overview • Lockheed-Martin Advanced Technologies Laboratories • RASSP Second Year Overview • RASSP Steering Committee -- Calendar of Events
4X - Charting the Course **Vol. 2, 2nd Qtr. 1995**	• Prime Development o Advance Technology Laboratories' Path to 4x Improvements o Road to 4x • Benchmark Program o RASSP Benchmark Program: Measuring 4x o Rapid Prototyping of Application-Specific Signal Processors: Current Practice, Challenges, and Roadmap • Technology Base

Issue	Table of Contents
	o Timing Insensitive Binary-to-Binary Translation (TIBBIT) o Graph Translation Tool (GrTT) • RASSP VHDL Working Group Update o RASSP Working Group Discusses Terms and Taxonomies
Very High Speed Integrated Circuits (VHSIC) **Hardware Description** **Language (VHDL)** **Vol. 2, 1st Qtr. 1995**	• Executive Outlook o RASSP and the Lockheed-Martin Merger o ARPA Manufacturing Technology Programs Ensure Military Access to Affordable Advanced Technology • Prime Development o VHDL Modeling for Signal Processor Development o Architectures for Rapid Prototyping of Embedded Signal Processors o Honeywell Develops VHDL Performance Model Library o Object-Oriented VHDL Provides New Modeling and Reuse Techniques for RASSP • Technology Base o The Ptolemy Kernel-Supporting Heterogeneous Design o VHDL Component Modeling: Impact on the RASSP Program • Benchmark Program o Assessing and Improving Current Practice in the Design Of Application-Specific Signal Processors • Editor's Corner o Editorial Viewpoint • Summer '95 RASSP Short Courses • Calendar of Events
RASSP After One Year **Vol. 1, 4th Qtr. 1994**	• RASSP After One Year • RASSP Education and Facilitation • Conceptual Prototype Demonstrates RASSP's Future • Martin Marietta RASSP Design Center Enables Design Environment Implementation • The Martin Marietta RASSP Team Demonstrates and Presents Rapid Prototyping Concepts at the First Annual Conference • Introduction to the Lockheed Sanders RASSP Team • The Lockheed Sanders RASSP Approach • The Lockheed Sanders Demonstration Program • Lockheed Sanders Beta Site Program • RASSP Conference Success • VHDL Models • The Benchmark 1 Executable Requirement • Vive La Difference • Acknowledgements • Available Technical Publications • RASSP Digest Rapid Prototyping of Application-Specific Signal Processors • RASSP Steering Committee • Calendar of Events

3.3 Courses, Seminars and Presentations

Educational and training material was developed that included course modules, seminars and presentations. Executive seminars consisted of slide presentations and information developed to bring information about RASSP technology to the managerial level in industry and government. At each of the seminars, input was sought and the feedback was used to update and improve the seminar presentations. Eighteen seminars were given on-site. A representative sample of seminar host sites is listed in Table 5.

Table 5 - Sites of Executive Seminars

Date	Organization	Number of Attendees
Nov-95	National Security Agency	50
Dec-95	NASA – Langley	35
Dec-95	Texas Instruments	35
Jan-96	MICOM	70
May-96	GEC Marconi	12
Sep-96	Rockwell Collins	12
Sep-96	Tinker ALC	10
Nov-96	Alliant Defense Systems	11

3.3.1 Course Modules

Course material was developed using a modular approach. Sixteen modules were developed under the program. Each module consists of an abstract of the material covered, lesson material, references, and in many cases a series of programmed questions and answers and a laboratory. These modules contain over 2,000 PowerPoint® slides.

Each module was designed to be a self-contained unit of instruction material, and provides the equivalent of three hours of classroom instruction. In addition, some modules include an additional three hours of lab work. A list of the modules appears in Table 6, and a complete abstract of each module is available in Appendix B. These modules plus two WAVES modules, that were developed as part of the U.S. Air Force's Fault Simulation and Test Pattern Generation program, are on the CD-ROM in Appendix C.

Table 6 - RASSP Modules

VHDL Basics
Structural VHDL
Behavioral VHDL
Advanced Concepts in VHDL
HW/SW Codesign Overview
DSP Architectures for RASSP
Scheduling and Assignment for DSP
DSP Algorithm Design
Communication Protocols
RASSP Methodology Overview
Requirements and Specification Modeling
Virtual Prototyping Using VHDL
Test Technology Overview
Cost Modeling for Embedded Digital Systems Design
Performance Modeling Using VHDL
Synthesis Using VHDL

The success of the approach is reflected in the demand for the use of these modules. Six university courses were offered by the RASSP E&F team member universities (UVA, GT, UC), and over 70 nonparticipating universities have used or are planning to use the RASSP modules.

The modules are designed to be adapted to different teaching environments. At some universities they have been used to facilitate the instruction of required engineering courses. Other examples of module usage in the classroom are listed in Table 7. This table does not include classes offered by the team member universities and shows how the modules are used outside of the RASSP E&F team. In addition, two modules were used as part of the lecture material for the interactive video network course "Multidisciplinarity and Collaborative Design for Systems on a Chip" taught by the Air Force Institute of Technology, Ohio State University and Oakland University.

Table 7 – Feedback on Modules Taught in Classes by Universities Not Associated with the RASSP E&F Team

University and Instructor	Description
Old Dominion University **- Dr. Jack Stoughton**	Used the VHDL modules to offer ECE695 Rapid Systems Prototyping - a first time VHDL course for graduate students. Modified modules to be used in a TV course. Plans to use VHDL module material to offer ECE443 Computer Architecture - a senior undergraduate required course.
University of Missouri, Rolla **- Dr. Hardy Pottenger**	Used the VHDL and Synthesis modules to teach a short course “Digital Logic Synthesis Using VHDL” to the IEEE St. Louis section. Used VHDL modules in EE311 Intro. to VLSI Design - a senior required course using FPGAs. Used RASSP Methodology Module in EE310 Intro. to Digital System Modeling Using VHDL - a senior undergraduate/entry level graduate course.

University and Instructor	Description
University of Notre Dame **- Dr. Jay Brockman**	Used VHDL modules in CSE322 Computer Architecture II - a senior level required course and CSE521 Graduate Computer Architecture - an entry level graduate course. Switched these courses from schematic-based design to top-down VHDL-based design. As a result, "students were able to do more and larger designs than was possible in previous years - writing behavioral descriptions first enabled errors to be caught earlier resulting in less redesign."
Loyola Marymount University **- Dr. Nazmul Ula**	Plans to introduce VHDL for the first time in the Fall of 1999 in ELEC698 VHDL Based Digital System Design - a senior undergraduate or graduate course. Future plans include introducing VHDL into the undergraduate logic design course and computer architecture course "We could not have introduced VHDL into the curriculum this early without the benefit of the VHDL modules."
Kansas State **- Dr. Bill Hudson**	Plans to use the VHDL modules in EECE241 Introduction to Computer Architecture - a beginning logic design course for freshman or sophomore undergraduates. Future plans are to introduce the use of VHDL, synthesis and virtual prototyping upwards in the curriculum as the class with first exposure moves up.
University of Massachusetts **- Dr. Wayne Burleson**	Used the DSP Algorithms and Architectures module in two graduate courses, VLSI Architectures and Computer Arithmetic. Plans to use other modules in a graduate course in Embedded Systems.
Cal. State University **- Dr. Larry Owens**	Plans to use VHDL lab material in ECE176 Computer Aided Engineering and Digital Design - a senior undergraduate elective course.

3.3.2 RASSP Workshop Participation

Members of the RASSP E&F team participated in several workshops and conferences by presenting papers about the RASSP program or making RASSP information and material available from booths. A representative list of these functions is shown in Table 8.

Table 8 - Selected Conferences and Workshops Having RASSP E&F Participation

- DSP World Conference
- IEEE Design Process Workshop
- DAC Conference
- VIUF
- IEEE VLSI DSP Workshop
- University of California, Berkeley
- 1st Annual RASSP Conference
- 2nd Annual RASSP Conference
- NASA System Engineering and Analysis Symposium'
- GOMAC
- Conference
- ASEE National Conference
- Mentor Users Group

Several VHDL Educators' Workshops were held which presented detailed information on RASSP and VHDL to educators. These workshops were a key component in helping the RASSP E&F team achieve their goal. By directing these workshops mainly at faculty members, it provided the opportunity to "teach the teacher" and thereby insert the technology into the academic environment at a quicker pace. A representative list of these workshops is shown in Table 9.

Table 9 - Selected Educator Workshops Provided by RASSP E&F

• VHDL Educator's Workshop • Digital Design 2000 - A Workshop on Innovations in System Design and Their Impact on Academic Curricula	• Top-Down Design of Embedded Digital Systems Educator's Workshop • IEEE Microelectronic Systems Education Conference • ARPA/VI Educator's Workshop

3.4 Publications

RASSP E&F has over 30 papers and/or presentations to its credit, including a special issue of *IEEE Design and Test of Computers: Rapid Prototyping* that focused on Rapid Prototyping and included both the RASSP technology and its Education and Facilitation. RASSP articles can be found in journals as well as conference proceedings. A representative listing of publications facilitated by RASSP with full citations can be found in the Bibliography.

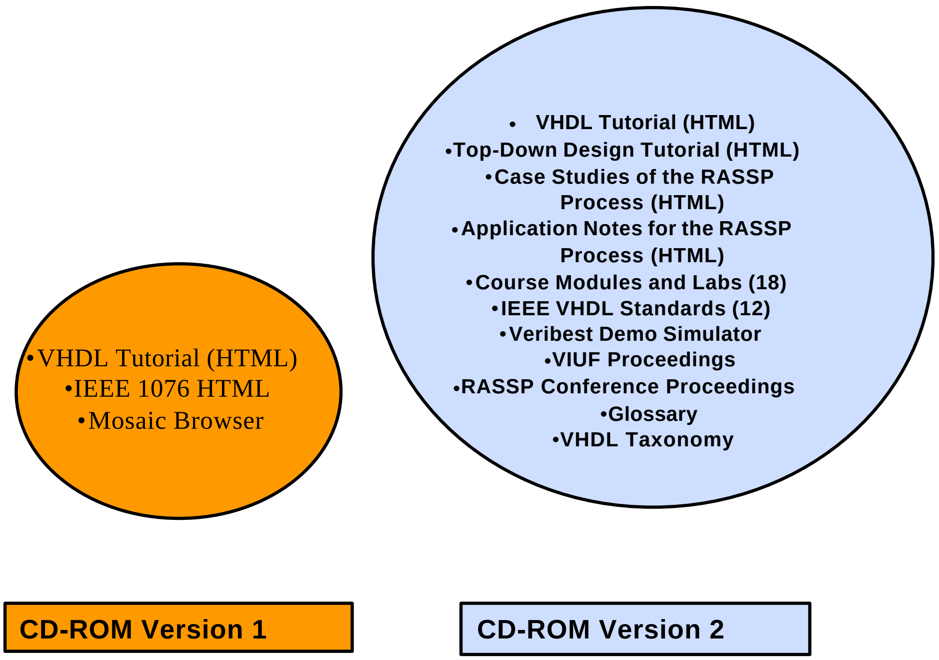
3.5 The RASSP CDs

Perhaps the pinnacle of the E&F program has been the success of the first RASSP CD-ROM, *VHDL Interactive Tutorial*, published by the IEEE. See Appendix C for the brochure. The CD-ROM provided the four VHDL course modules in HTML format linked to the VHDL Language Reference Manual IEEE 1076-1993 and Mosaic Browsers (see Figure 13). Between March 1997 and December 1999, 816 copies were sold. The CD was available as stand alone or bundled with IEEE standards.

RASSP CD Sales 1st Edition:

506 CD only sales
310 CD with standards sales

Figure 13 - RASSP CD-ROM


•VHDL Tutorial (HTML)
•IEEE 1076 HTML
•Mosaic Browser
• VHDL Tutorial (HTML)
•Top-Down Design Tutorial (HTML)
•Case Studies of the RASSP Process (HTML)
•Application Notes for the RASSP Process (HTML)
•Course Modules and Labs (18)
•IEEE VHDL Standards (12)
•Veribest Demo Simulator
•VIUF Proceedings
•RASSP Conference Proceedings
•Glossary
•VHDL Taxonomy
CD-ROM Version 1
CD-ROM Version 2

The second edition of the CD-ROM, *VHDL: Electronic Systems Design Methodologies and Interactive Tutorial*, was renamed so that the title encompassed the additional material (see Figure 13). The second edition contains 50 times the information of the first CD-ROM. It also has over 500 Megabytes of information in over 8,800 files with over 6,000 hyperlinks. The CD-ROM is a multi-faceted presentation mechanism and provides many links between its different focus areas. The CD-ROM organization allows the user to start in any of these focus areas, thereby providing maximum utility. As shown in the table of contents (Table 10), the second edition of the CD-ROM provides information for those starting in VHDL as well as those looking to improve their top-down design and VHDL knowledge. Some of the material comes from other programs such as the U.S. Air Force VHDL Interactive Training for Acquisition and Maintenance Specialists (VITAMINS) program under contract number F33615-96-C-1910 and U.S. Air Force, Fault Simulation and Test Pattern Generation program under contract 95-C-0220. To help get this information into the academic environment, the RASSP E&F program in cooperation with the IEEE sent a copy of the CD-ROM to the electrical engineering department heads of 107 universities and colleges in the United States. A list of these schools is provided in Table 11. Much of the RASSP generated material on the

CD-ROM will be placed on the RASSP webserver, which will remain operational for a period after the program ends.

Table 10 - Table of Contents for Second RASSP CD-ROM

VHDL: Electronic Systems Design Methodologies and Interactive Tutorial

- **VHDL Tutorial -** HTML sequence of course modules 10, 11, 12, 13 & 60 and labs for Unix and Windows systems.
- **Top-Down Design -** HTML sequence of course modules 14, 29, 30, 32, 57 & 59 and labs for Unix and Windows systems.
- **Case Studies -** These real world benchmark case studies provide an introduction to top-down design process. Even though embedded Digital Signal Processors (DSP) are used as delivery mechanisms for the process, it can be applied in other areas.
- **Application Notes -** These application notes provide additional detail in selected technical areas that are covered in the Case Studies. They can also be uses as stand-alone references.
- **VHDL Based Acquisition -** Material from the VITAMINS program which covers the use of top-down design in developing requirements and performing system design.
- **Course Modules -** These are 18 PPT course modules, which were designed to be taught in undergraduate/graduate courses. In addition, two labs covering the complete top-down process are provided. Includes two modules from the Fault Simulation and Test Pattern Generation program.
- **VHDL Standards -** The key IEEE VHDL standards and a military handbook for VHDL design.
  - IEEE Std 1029.1-1998, IEEE Standard for VHDL Waveform and Vector Exchange (WAVES) to Support Design and Test Verification
  - IEEE Std 1076-1993, IEEE Standard VHDL Language Reference Manual
  - IEEE Std 1076a-2000, Amendment to IEEE Std 1076-1993
  - IEEE Std 1076/INT-1991, IEEE Standards Interpretations: IEEE Std 1076-1987 IEEE Standard VHDL Language Reference Manual
  - IEEE Std 1076.1-1999, IEEE Standard VHDL Analog and Mixed-Signal Extensions
  - IEEE Std 1076.2-1996, IEEE Standard VHDL Mathematical Packages
  - IEEE Std 1076.3-1997, IEEE Standard VHDL Synthesis Packages
  - IEEE Std 1076.4-1995, IEEE Standard for VITAL Application-Specific Integrated Circuit (ASIC) Modeling Specification
  - Approved Draft of IEEE Std 1076.6-1999, IEEE Standard for VHDL Register Transfer Level Synthesis
  - IEEE Std 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture
  - IEEE Std 1149.1b-1994, Supplement to IEEE Std 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture
  - IEEE Std 1164-1993, IEEE Standard Multivalue Logic System for VHDL Model Interoperability (Std_logic_1164)
  - MIL-HDBK 62, Documentation of Digital Electronic Systems with VHDL
- **References -** When possible, references listed in the material are included on the CD-ROM. This includes RASSP material as well as VIUF Conference Proceedings.
- **VHDL Models -** These are models that were developed as part of the RASSP program. They include process and memory models.
- **VHDL Tools -** These include a VeriBest simulator, VHDL GUI interface and VHDL model generator.
- **VHDL Taxonomy -** This document provides a framework for classifying VHDL models to promote model reuse.
- **Glossary -** Provides definitions of terms within the context of the VHDL language, VHDL simulation, Digital Signal Processing and RASSP. Links to documents where the terms are used are included.

Table 11 - List of Schools that have Received the 2nd Edition of the RASSP CD-ROM

SCHOOL	DEPT.
AIR FORCE INSTITUTE OF TECHNOLOGY	Dept. of Electrical and Computer Engineering
CATHOLIC UNIVERSITY OF AMERICA	Dept. of Electrical Engineering
CHRISTIAN BROTHERS UNIVERSITY	Dept. of Electrical Engineering
CITADEL MILITARY COLLEGE	Dept. of Electrical Engineering
CLARKSON UNIVERSITY	Dept. of Electrical and Computer Engineering
CLEMSON UNIVERSITY	Dept. of Electrical and Computer Engineering
CLEVELAND STATE UNIVERSITY	Dept. of Electrical and Computer Engineering
COLLEGE OF NEW JERSEY	Dept. of Engineering
COLORADO STATE UNIVERSITY	Dept. of Electrical Engineering
COOPER UNION	Dept. of Electrical Engineering
DARTMOUTH COLLEGE	Thayer School of Engineering
DREXEL UNIVERSITY	Dept. of Electrical and Computer Engineering
EMBRY-RIDDLE AERONAUTICAL UNIVERSITY	Dept. of Electrical Engineering and Computer Science
FAIRLEIGH DICKINSON UNIVERSITY	School of Engineering & Engineering Technology
FLORIDA AGRICULTURAL & MECHANICAL UNIVERSITY	Dept. of Electrical Engineering
FLORIDA ATLANTIC UNIVERSITY	Dept. of Electrical Engineering
FLORIDA INSTITUTE OF TECHNOLQGY	Division of Electrical & Computer Science and Engineering
GANNON UNIVERSITY	Dept. of Electrical Engineering
GEORGE MASON UNIVERSITY	Dept. of Electrical and Computer Engineering
GONZAGA UNIVERSITY	Dept. of Electrical Engineering
GROVE CITY COLLEGE	Dept. of Electrical Engineering
HAMPTON UNIVERSITY	Dept. of Electrical Engineering
HARVEY MUDD COLLEGE	Dept. of Engineering
HOFSTRA UNIVERSITY	Dept. at Electrical Engineering
HOWARD UNIVERSITY	Dept. at Electrical Engineering
KETTERING UNIVERSITY	Electrical and Computer Engineering Dept.
LAFAYETTE COLLEGE	Dept. of Electrical Engineering
LAMAR UNIVERSITY	Dept. of Electrical Engineering
LAWRENCE TECHNOLOGICAL UNIVERSITY	Dept. of Electrical Engineering
LEHIGH UNIVERSITY	Dept. of Electrical Engineering & Computer Science
LOUISIANA TECH UNIVERSITY	Dept. of Electrical Engineering
LOYOLA MARYMOUNT UNIVERSITY	Dept. of Electrical Engineering and Computer Science
MANHATTAN COLLEGE	Dept. of Electrical Engineering
MANKATO STATE UNIVERSITY	Electrical Engineering & Electronic Engineering
MERCER UNIVERSITY	Dept. of Electrical & Computer Engineering
MERRIMACK COLLEGE	Electrical/Computer Engineering Dept.
MILWAUKEE SCHOOL OF ENGINEERING	Dept. of Electrical Engineering & Computer Science
MISSISSIPPI STATE UNIVERSITY	Dept. of Electrical & Computer Engineering
MONMOUTH UNIVERSITY	Electronic Engineering Dept.
MONTANA STATE UNIVERSITY	Dept. of Electrical & Computer Engineering
MORGAN STATE UNIVERSITY	Dept. of Electrical Engineering
NAVAL POSTGRADUATE SCHOOL	Dept. of Electrical & Computer Engineering

SCHOOL	DEPT.
NEW JERSEY INSTITUTE OF TECHNOLOGY	Electrical & Computer Engineering Dept.
NEW MEXICO STATE UNIVERSITY	The Klipsch School of Electrical & Computer Engineering
NORTH CAROLINA A&T STATE UNIVERSITY	Dept. of Electrical Engineering
NORTH DAKOTA STATE UNIVERSITY	Electrical Engineering Dept.
NORTHEASTERN UNIVERSITY	Dept. of Electrical & Computer Engineering
NORTHWESTERN UNIVERSITY	Dept. of Electrical & Computer Engineering
NORWICH UNIVERSITY	Dept. of Electrical Engineering
OAKLAND UNIVERSITY	Dept. of Electrical & Systems Engineering
OHIO NORTHERN UNIVERSITY	Dept. of Electrical Engineering
OHIO STATE UNIVERSITY	Dept. of Electrical Engineering
OKLAHOMA STATE UNIVERSITY	Dept. of Electrical & Computer Engineering
POLYTECHNIC UNIVERSITY	Dept. of Electrical Engineering
PRAIRIE VIEW A&M UNIVERSITY	Dept. of Electrical Engineering
SAGINAW VALLEY STATE UNIVERSITY	Electrical & Computer Engineering Dept.
SAINT LOUIS UNIVERSITY	Dept. of Electrical Engineering
SEATTLE PACIFIC UNIVERSITY	Dept. of Electrical Engineering
SEATTLE UNIVERSITY	Dept. of Electrical Engineering
SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY	Dept. of Electrical & Computer Engineering
SOUTH DAKOTA STATE UNIVERSITY	Dept. of Electrical Engineering
ST. CLOUD STATE UNIVERSITY	Dept. of Electrical Engineering
TEMPLE UNIVERSITY	Dept. of Electrical & Computer Engineering
TENNESSEE TECHNOLOGICAL UNIVERSITY	Dept. of Electrical & Computer Engineering
TEXAS A&M UNIVERSITY - KINGSVILLE	Dept. of Electrical Engineering & Computer Science
TEXAS TECH UNIVERSITY	Dept. of Electrical Engineering
TRINITY COLLEGE	Dept. of Engineering
TRINITY UNIVERSITY	Dept. of Engineering Science
TRI-STATE UNIVERSITY	Dept. of Electrical & Computer Engineering
TUSKEGEE UNIVERSITY	Dept. of Electrical Engineering
UNITED STATES AIR FORCE ACADEMY	Dept. of Electrical Engineering
UNITED STATES COAST GUARD ACADEMY	Dept. of Engineering
UNITED STATES MILITARY ACADEMY	Dept. of Electrical Engineering & Computer Science
UNITED STATES NAVAL ACADEMY	Electrical Engineering Dept.
UNIVERSITY OF AKRON	Dept. of Electrical Engineering
UNIVERSITY OF ALABAMA AT BIRMINGHAM	Dept. of Electrical & Computer Engineering
UNIVERSITY OF ALABAMA IN HUNTSVILLE	Dept. of Electrical & Computer Engineering
UNIVERSITY OF ALASKA - FAIRBANKS	Electrical Engineering Dept.
UNIVERSITY OF BRIDGEPORT	Dept. of Electrical Engineering
UNIVERSITY OF CENTRAL FLORIDA	Dept. of Electrical & Computer Engineering
UNIVERSITY OF DAYTON	Dept. of Electrical & Computer Engineering
UNIVERSITY OF EVANSVILLE	Dept. of Electrical Engineering & Computer Science
UNIVERSITY OF HAWAII AT MANOA	Dept. of Electrical Engineering
UNIVERSITY OF KENTUCKY	Dept. of Electrical Engineering

SCHOOL	DEPT.
UNIVERSITY OF MAINE	Dept. of Electrical & Computer Engineering
UNIVERSITY OF MARYLAND AT BALTIMORE COUNTY	Dept. of Computer Science & Electrical Engineering
UNIVERSITY OF MINNESOTA, DULUTH	Dept. of Electrical & Computer Engineering
UNIVERSITY OF NEBRASKA	Dept. of Electrical Engineering
UNIVERSITY OF NEVADA, LAS VEGAS	Dept. of Electrical & Computer Engineering
UNIVERSITY OF NEW HAMPSHIRE	Dept. of Electrical & Computer Engineering
UNIVERSITY OF NEW HAVEN	Dept. of Electrical & Computer Engineering
UNIVERSITY OF NEW ORLEANS	Dept. of Electrical Engineering
UNIVERSITY OF NORTH CAROLINA AT CHARLOTTE	Electrical Engineering Dept.
UNIVERSITY OF NORTH DAKOTA	Dept. of Electrical Engineering
UNIVERSITY OF PUERTO RICO - MAYAGUEZ	Electrical & Computer Engineering Dept.
UNIVERSITY OF SOUTH ALABAMA	Dept. of Electrical & Computer Engineering
UNIVERSITY OF SOUTH CAROLINA	Dept. of Electrical & Computer Engineering
UNIVERSITY OF SOUTHERN MAINE	Dept. of Engineering
UNIVERSITY OF SOUTHWESTERN LOUISIANA	Dept. of Electrical & Computer Engineering
UNIVERSITY OF TENNESSEE, KNOXVILLE	Dept. of Electrical Engineering
VALPARAISO UNIVERSITY	Electrical & Computer Engineering Dept.
VIRGINIA MILITARY INSTITUTE	Dept. of Electrical Engineering
WEBB INSTITUTE OF NAVAL ARCHITECTURE	Dept. of Electrical Engineering
WEST VIRGINIA UNIVERSITY	Dept. of Computer Science & Electrical Engineering
WEST VIRGINIA UNIVERSITY INSTITUTE OF TECHNOLOGY	Dept. of Electrical Engineering
WESTERN MICHIGAN UNIVERSITY	Dept. of Electrical & Computer Engineering
WESTERN NEW ENGLAND COLLEGE	Dept. of Electrical Engineering
WIDENER UNIVERSITY	Dept. of Electrical Engineering
WILLIAM MARSH RICE UNIVERSITY	Dept. of Electrical & Computer Engineering

4 LESSONS LEARNED

It soon became obvious that the lifeline and the success of the RASSP E&F program was going to depend on the ability of RASSP E&F to obtain information from the other RASSP programs. Unlike most programs, companies needed to share their knowledge to improve the overall design process. Initially, some of the companies involved with the RASSP program where reluctant to provide the level of detail needed to understand and implement the companies' findings. As these companies became more comfortable talking with E&F team members, and as information from other RASSP program members became available, information from the companies flowed more freely. Help from the government managers was key in breaking down barriers that naturally occur in the competitive environment of electronic design.

The availability of RASSP information to the public was an important key to its success. When RASSP started, the World Wide Web was in its infancy and as the web became more prevalent, access by both industry and academia increased. The RASSP web server became the dominant passive mechanism for providing information.

Publication of the *RASSP Digest* provided a formal approach for providing information to managers, engineers and professors. RASSP participants wrote the articles describing their tasks and accomplishments. Many times this provided information that would not have otherwise been published.

The RASSP E&F philosophy of "teaching the teacher" shortened the time required for new/different methodologies to make inroads into the university environment. The workshops and seminars where a critical component of reaching the academic community. The modular approach that we took allowed us not only to educate the teachers, but also to provide them with the support material (modules) needed to quickly integrate what was learned into their lectures.

Leading professors in the electronic design field developed the RASSP E&F modules. Based on their knowledge and experience, they were able to distill the information collected from the other RASSP participants into a form that was understandable and usable by the academic community. However, one problem that was encountered was that the effort to gather the material, analyze it, create an outline and then produce a quality slide was long and difficult. Slide generation was especially a problem because each slide and accompanying note page was created manually by engineers with expertise in the subject area. Non-technical individuals performed only slide formatting. Engineers also reviewed the slides and performed additional editing. While this required technical resources to perform some non-technical tasks, it was deemed to be the best approach. Much of the material was created from scratch or from handwritten course notes and to put the material in a form in which non-technical individuals could produce the slides would have added additional steps to the process.

Summary

Significant accomplishments of the RASSP E&F program include:

- The establishment and maintenance of a webserver;
- The publication of the *RASSP Digest*;
- Use of RASSP E&F modules by over 70 engineering schools;
- Development of educational and training materials and implementation of technology transfer including:
- Executive Seminars;
- Workshops;
- RASSP Course Modules; and
- IEEE publication of two RASSP CD-ROMs.

The results of the RASSP E&F can be summarized as follows:

RASSP E&F set off as an unprecedented program set up to disseminate the information developed by 24 other RASSP programs to enable a paradigm shift in the way signal processors are designed.

With more than 30 papers and/or presentations to its credit, RASSP has been able to reach a wide national audience. The website has been accessed by dozens of universities and hundreds of commercial enterprises, indicating the nature and far ranging impact of RASSP E&F. These accomplishments indicate the successful achievement of generating awareness and eliciting user interest, as well as educating industry on the benefits of RASSP technology.

Of the 264 engineering schools in the United States which grant EE degrees, RASSP E&F interacted with over 100 of these. Over 70 schools have used modules developed by RASSP E&F. In addition by providing 107 schools that offer electrical engineering programs with a copy of the 2nd edition of the RASSP CD-ROM, the RASSP methodologies will be available to hundreds of professors.

The second edition of the RASSP CD-ROM presents the essence of the knowledge from the entire RASSP program. This knowledge is available not only to professors and students, but also to practicing engineers that may want to learn new skills or to enhance their knowledge in areas in which they are already working. Because of the broad spectrum of effort (executive seminars, workshops, course modules, CD-ROMS, webpage) RASSP E&F surpassed its goal of developing, incorporating and disseminating an education system to teach state-of-the-art system design techniques.

5 BIBLIOGRAPHY

1. IEEE Design and Test of Computers: Rapid Prototyping, Fall 1996.

2. Batchman, T., ed., IEEE Transactions on Education, Vol. 40, No. 15, 1997.

3. King, S.Y., ed., Journal of VLSI Signal Processing Systems, Vol. 15, Nos. 1 & 2, 1997.

4. Gadient, A., Hines, L., Welsh, J., Schwalb, A., "The RASSP Manufacturing Interface: Enabling Distributed Design Through an Agile Infrastructure, Results from a Production Implementation," *Concurrent Engineering Research and Applications Journal,* Spring 1997.

5. Gadient, A.J., Madisetti, V.K., Aylor, J.H., Wilsey, D.P.,"A Paradigm Shift in Digital System Design Education With Industry Participation," *Proceedings American Society of Engineering Educators Annual Conference,* Washington, D.C, June 1996.

6. Bullock, K, Streeter, M., Hoffman, G., Muncaster, McCullough, M., "Managing the RASSP Virtual Enterprise", Proceedings of Second Annual RASSP Conference, Arlington, VA, July 1995.

7. Hein, C., "Exploiting VHDL Design in RASSP," Published in *Proceedings of VHDL International Users' Forum*, Fall 1994 Conference, McLean, VA, November 1994.

8. Hein, C., Gadient, A., *et al.*, "A VHDL Modeling Terminology and Taxonomy for RASSP," Published in *Proceedings VHDL International User's Forum*, Boston, MA, October 1995.

9. Freedman, J., ed., "Design Needs for the 21st Century," SRC White Paper, September 1994.

10. Rundquist, "Model Year Upgrade of the AN/AAS-42 Infrared Search and Track", RASSP Digest, Vol. 2, 4th Qtr. 1995.

11. Smerdon, E.T., "Lifelong Learning for Engineers: Riding the Whirlwind," National Academy of Engineering Workshop, June 27, 1996.

12. Zebrowitz, H., "The Rapid-Prototyping Application-Specific Signal Processors (RASSP) Program," *IEEE Spectrum,* July 1996, p. 81.

13. Saultz, J.E., "Rapid Prototyping of Application-Specific Signal Processors (RASSP) In-Progress Report," *Journal of VLSI Signal Processing*, Vol.15, No. 1, January 1997.

14. Pridmore, J., "The Standard Virtual Interface - An Interoperability Approach", *RASSP Digest*, Vol. 2, 4th Qtr. 1995.

15. Welsh, J., Kalathil, B., Chadha, B., Finnie, E., Tuck, M., Selvidge, W., and Bard, A., "Integrated Process Control and Data Management in RASSP Enterprise Systems", Published in *Proceedings of the Second Annual RASSP Conference*, Arlington, VA, July 1995.

16. Scanlan, L., Lee, W., Vahey, M., and McCollough, M., "RASSP Methodology Evaluation and Lessons Leaned Developing IRST Signal Processor, *Journal of VLSI Signal Processing*, Vol.15, No. 2, January 1997.

17. Scanlan, L. and Fisher, L., “Road to 4X”, RASSP Digest, Vol. 2, 2nd Qtr. 1995.

18. Richards, M.A, Gadient, A.J., Frank, G.A., and Harr R., “The RASSP Program: Origin, Concepts, and Status”, *Journal of VLSI Signal Processing*, Vol.15, No. 1, January 1997.

19. Richards, M., “The RASSP Program Overview and Accomplishments”, Proceedings of First Annual RASSP Conference, Arlington, VA, August 1994.

20. Richards, M.A, Gadient, A.J., and Frank, G.A., eds., *Rapid Prototyping of Application Specific Signal Processors*, Kluwer Academic Publishers, February, 1997.

21. Madisetti, V., Corley, and Shaw, G., “Rapid Prototyping of Application Specific Signal Processors - Educator/Facilitator Current Practice Model (1993) and Challenges”, *Proceedings of Second Annual RASSP Conference,* Arlington, VA, July 1995.

APPENDIX A -RASSP Program Participants

Government Efforts

Air Force
Army
DARPA
Navy

Prime Efforts

Lockheed Martin Advance Technology Laboratory
Sanders, A Lockheed Martin Company

Technology Base Efforts

CFI
Georgia Institute of Technology
Honeywell Technology Center
Hughes Radar and Communication Systems
Intermetrics, Inc.
JRS Research Laboratories, Inc.
Management Communications and Control, Inc.
Massachusetts Institute of Technology
Mercury Computer Systems, Inc.
Mississippi State University
Ohio State University
Omniview, Inc.
University of California at Berkeley
University of California at Davis
University of Cincinnati
University of Minnesota
University of Oregon
University of Virginia

Benchmarker Effort

Massachusetts Institute of Technology Lincoln Laboratory

Educator and Facilitator Effort

SCRA's Advanced Technology Institute

APPENDIX B -Abstract Descriptions Of The RASSP E&F Course Modules

- ## Module 10 -- VHDL Basics
  - **Abstract:** The VHDL Basics module introduces the VHSIC Hardware Description Language and its fundamental concepts. VHDL is a language specifically developed to describe digital electronic hardware and its attributes. VHDL is a flexible language that can be applied to many different design situations. This language has several key advantages, including technology independence and a standard language for communication. The module describes many of the advantages of using VHDL and a short history of the language.

- ## Module 11 -- Structural VHDL
  - **Abstract:** The Structural VHDL module describes the use of VHDL for describing models in terms of component instantiations and interconnections. Structural VHDL can be appropriate at any level of design. For example, testbenches for completed components are often described using structural VHDL. Furthermore, structural VHDL supports the use of libraries and component reuse.

- ## Module 12 -- Behavioral VHDL
  - **Abstract:** The Behavioral VHDL module describes features of the language that describe the behavior of components in response to signals. Behavioral descriptions of hardware utilize software engineering practices and constructs to achieve a functional model. Timing information is not necessary in a behavioral description, although such information certainly can be added.

- ## Module 13 -- Advanced Concepts in VHDL
  - **Abstract:** The Advanced Concepts in VHDL module spans a wide range of topics, including several that may be applied to higher levels of design abstraction. Many of these constructs will have been introduced in the first three VHDL modules in this sequence, but this module covers them more comprehensively. Examples of such constructs include signal assignment statements, and the capabilities and differences when they are used as concurrent VHDL statements or sequential VHDL statements. Similarly, the VHDL process is discussed in more detail than in earlier modules. It should also be noted that TEXTIO and shared variables are introduced in this module.

- ## Module 14 -- Hardware/Software Codesign Overview
  - **Abstract:** The Hardware/Software Codesign Overview module is intended to introduce the hardware/software codesign to the practicing design, software, and systems engineers, and to the senior undergraduate or first year graduate student.

The module provides key codesign concepts and attempts to show the benefits of the codesign approach over the current design process.

- ## Module 21 -- DSP Architectures for RASSP

  - **Abstract:** The DSP Architectures for RASSP module is intended to provide digital signal processing (DSP) architectures both from an historical and an RASSP perspective to system and architecture design engineers or to first year graduate students.

- ## Module 22 -- Scheduling and Assignment for DSP

  - **Abstract:** The Scheduling and Assignments for DSP module is intended to provide to system engineers or to first year graduate students an understanding of scheduling and assignment concepts with emphasis on DSP applications.

- ## Module 23 -- DSP Algorithm Design

  - **Abstract:** The DSP Algorithm Design module is intended to provide to system engineers or to first year graduate students an understanding of DSP algorithm design.

- ## Module 25 -- Communication Protocols

  - **Abstract:** The Communication Protocols module is intended to provide to system engineers or to first year graduate students an understanding of communications in scalable DSP architectures.

- ## Module 29 -- RASSP Methodology Overview

  - **Abstract:** The RASSP Methodology Overview module provides an introduction to how design practice is studied and how improved methodology is implemented and continuously refined. Definitions are provided so that a consistent terminology is established. The module gives a comparison of the pre-RASSP and current RASSP methodology. It also describes potential process improvements and how they may enable the RASSP program to achieve its cost and life cycle reduction goals. Examples of key improvements such as hardware/software codesign, integrated product development, enterprise integration and virtual prototyping are described. Finally, the module shows the enterprise integration mechanisms used to control and manage design methodology.

- ## Module 30 -- Requirements and Specifications Modeling
  - **Abstract:** The Requirements and Specification (RSM) module provides an introduction to the topic of executable requirements and specifications. Their use leads toward a more formalized listing of requirements and specifications than has been traditionally provided.

- ## Module 32 -- Virtual Prototyping using VHDL
  - **Abstract:** In today's engineering design environment, designers are limited in their ability to maximize reuse by the fact that there is no efficient way to search for, access, and integrate reusable design objects across multiple sources; frequently, these potential sources of reusable design data are uncoupled from the design environment. This paper details an approach for managing reusable design objects in a collaborative engineering environment that enables Rapid Prototyping of Application-Specific Signal Processors (RASSP) and the architecture of the RASSP Reuse Data Manager (RRDM), specifically developed to support this approach.

- ## Module 43 -- Test Technology Overview
  - **Abstract:** The Test Technology Overview module is intended to provide an overview of digital systems testing to the general design engineer. The module contains basic information on the fundamentals of testing including motivation, current practice, and basic fault modeling techniques. The basic algorithms for test generation and fault simulation for both combinational and sequential designs are then covered followed by a presentation of the theory of IDDQ testing.

- ## Module 57 -- Cost Modeling for Embedded Digital Systems Design
  - **Abstract:** Designers of high-end embedded systems or large volume consumer products are faced with the challenge of rapidly prototyping designs which meet stringent electrical specifications along with tight physical constraints, under restrictive system engineering constraints such as cost time to market (TTM) and resource limitations. The goal is to design a minimum cost system, with consideration of lifecycle costs, as opposed to a minimum cost hardware system. This module describes a new RASSP design methodology called Cost Modeling and its application to the embedded digital system design process. A detailed case study and a thorough description of hardware and software cost estimators are presented.

- ## Module 59 -- Performance Modeling using VHDL
  - **Abstract:** The Performance Modeling Using VHDL module is intended to present the area of system level performance modeling using VHDL. The first section of the module includes a background of performance modeling including the objectives of performance modeling and definitions of common performance modeling terms. Techniques for performance modeling such a Petri Nets, queuing models, and uninterpreted models are covered along with how simulation based performance modeling is implemented in VHDL.

- ## Module 60 -- Synthesis using VHDL
  - **Abstract:** The Synthesis using VHDL module describes how one can synthesize digital systems using VHDL. It does not teach VHDL, nor does it teach synthesis. The former is the task of earlier modules, while the latter is the task of various synthesis tools that take in an input specification in VHDL and process it.

APPENDIX C - *VHDL: Electronic Systems Design Methodologies and Interactive Tutorial*

Published by the IEEE (brochure and limited version of the CD-ROM.)

This version CD-ROM contains only the RASSP components of the complete publication.

IEEE Standards Information Network

IEEE Standards Activities
445 Hoes Lane
PO Box 1331
Piscataway NJ 08855-1331 USA

http://standards.ieee.org/

A

the comprehension and use of IEEE **VHDL** *(Very High Speed Integrated Circuit Hardware Description Language)*, this unique and sophisticated product offers a comprehensive and reliable tutorial on **VHDL** and **Top-Down Design** only available from IEEE.

T

provides fully interactive **VHDL** and **Top-Down Design** tutorials with links to a hyper-text version of IEEE Std 1076-1993. These robust tutorials are supported by a vast quantity of "real-world" based Case Studies and Application Notes, based on the DARPA Rapid Prototyping of Application Specific Signal Processors (RASSP) program. In addition to these, laboratory exercises, **VHDL** Models, **VHDL** Tools, references and the complete set of viewable **VHDL** standards are on the CD-ROM which provides over 500 MB of user-friendly and comprehensive information.

VHDL teractive utorial

dows 5,'98

A

In the computer systems industry, incompatible and often proprietary electronic design description languages were once used for describing hardware making it difficult for engineers to understand design documentation. Proprietary description languages hindered hardware development and created the need for a uniform, industry-accepted means of describing hardware. IEEE responded in 1987 by standardizing the **VHSIC** *(Very High Speed Integrated Circuit)* **Hardware Description Language**, or **VHDL**.

VHDL is a broad-based formal notation implemented throughout today's industry in all phases of electronic systems development. This is evidenced by the variety of **VHDL** tools available and used worldwide.

eadable by both machines and humans, **VHDL** supports the...

- Development, verification, synthesis and testing of hardware design
- Communication of hardware design data
- Maintenance, modification and procurement of hardware

T

VHDL Interactive Tutorial CD-ROM integrates many different focus areas through a sophisticated link structure. The CD-ROM organization allows you to start in any of these focus areas, thereby providing maximum utilization.

These focus areas include:

- VHDL Tutorial
- Top-Down Design Tutorial
- Case Studies
- Application Notes
- VHDL-Based Acquisition
- Course Modules
- VHDL Standards
- References
- VHDL Models
- VHDL Tools
- VHDL Taxonomy
- Glossary

This product is licensed for single-user purposes. Use for multiple-user licenses will be available upon request. E-mail such requests to stds-vhdlcdrom@ieee.org.

See the "README.TXT" file on the CD-ROM for operation instructions.

ORDER FORM

Description	Product No.	List Price	Member Price	Qty.	Total
VHDL: *Electronic Systems Design Methodologies and Interactive Tutorial*	SP1120-NCE	$140.00	$112.00 *MEMBER PRICE ON THE FIRST COPY OF PRODUCT ORDERED		

Subtotal	
DC, FL, NJ, MI, OH Shipments Add Sales Tax	
Handling Charge	
CA, NY shipments, add Sales Tax on Subtotal and Handling	
Residents of Canada, add appropriate GST/HIS tax	
TOTAL	

HANDLING CHARGES

$50 or less	$ 4
$50.01-$75	$ 5
$75.01-$100	$ 6
$100-$200	$ 8
$200+	$15

IEEE Canadian Business Number 125634188

Name ______________________ IEEE or IEEE-SA Member Number ______________

Title ______________________ Company ______________________

Address ____

City ______________ State/Prov. ______ Zip/Postal Code ________ Country ______

Email ____

Phone ______________________ Fax ______________________

Please check the appropriate box:

❐ Payment enclosed (make checks payable to IEEE)

❐ Please invoice me (subject to credit approval; purchase order required)

❐ Please charge to the credit card listed below ❐ AMEX ❐ MasterCard/EuroCard ❐ Diner's Club ❐ VISA

Card No. ______________________________ Exp. Date ________

Signature ______________________________

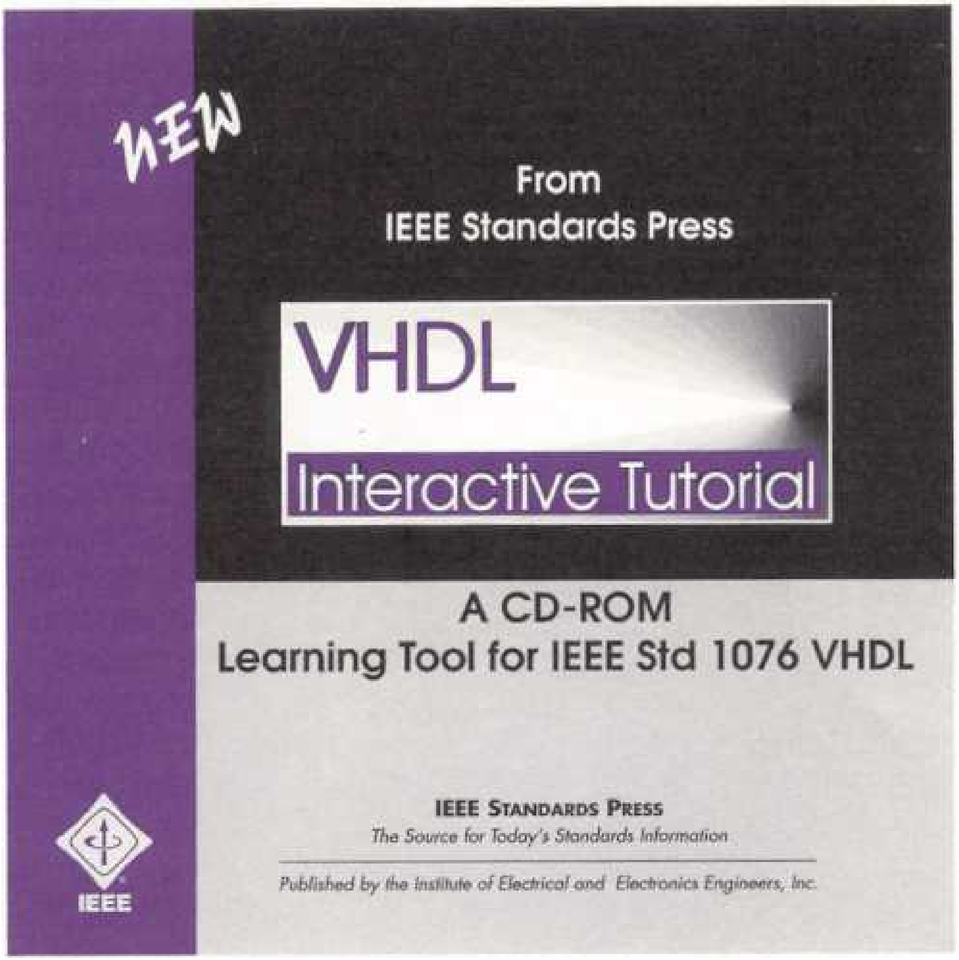
CONTACT/ORDER INFORMATION

Mail: IEEE Standards Information Network
IEEE Standards Activities
445 Hoes Lane
PO Box 1331
Piscataway NJ 08855-1331 USA

URL: http://standards.ieee.org/
Phone: +1 800 678 IEEE (4333) (in the US & Canada)
+1 732 981 0060 (outside of the US & Canada)
Fax: +1 732 981 9667
Email: customer.service@ieee.org

APPENDIX D - *VHDL Interactive Tutorial*

Published by the IEEE (brochure only.)



VHDL Interactive Tutorial

Your complete VHDL training tool

Aiding in the comprehension and use of IEEE VHDL (Very High Speed Integrated Circuit Hardware Description Language), this unique product offers a comprehensive and reliable tutorial on VHDL—not available anywhere else. An enhancement to IEEE Std 1076-1993, the interactive tutorial is organized into four modules designed to incrementally add to the users' understanding of VHDL and its applications. By integrating these modules* with the *VHDL Language Reference Manual*, IEEE Std 1076-1993, in a hypertext environment, this outstanding teaching tool helps users learn the language and makes VHDL more usable.

This hands-on tutorial shows clear links between the many levels and layers of VHDL and provides actual examples of VHDL implementation, making it an indispensable tool for VHDL product developers and users. It describes the construct of the VHDL interface specs—what VHDL is, what it does and how it's implemented.

The VHDL CD-ROM tutorial provides an easy-to-use logical method of referencing the standard. Moreover, it comes bundled with the Spyglass® Mosiac™ 2.11 browser and is available for use in Windows™ (3.1 and 95), Macintosh®, Sun® OS and Sun Solaris® environments.

VHDL Interactive Tutorial CD-Rom
Available in the Following popular formats

- Microsoft Windows® (3.1 and '95 compatible)
- Macintosh®
- Sun® OS
- Sun Solaris®

Each format is also available packaged with a printed copy of IEEE Std 1076-1993, *VHDL Language Reference Manual*. See order form for details.

About VHDL . . .

In the computer systems industry, incompatible and often proprietary electronic design description languages were once used for describing hardware — making it difficult for engineers to understand design documentation. Proprietary description languages hindered hardware development and created the need for a uniform, industry-accepted means of describing hardware. IEEE responded in 1987 by standardizing the VHSIC (Very High Speed Integrated Circuit) Hardware Description Language, or VHDL.

VHDL is a broad-based formal notation implemented throughout today's industry in all phases of electronic systems development. This is evidenced by the variety of VHDL tools available and used worldwide.

Readable by both machines and humans, VHDL supports the

- Development, verification, synthesis and testing of hardware design
- Communication of hardware design data
- Maintenance, modification and procurement of hardware

Four Easy-to-Use Modules

Organized into four easy-to-use modules that build upon each other, the VHDL Interactive Tutorial allows users to set their own learning pace.

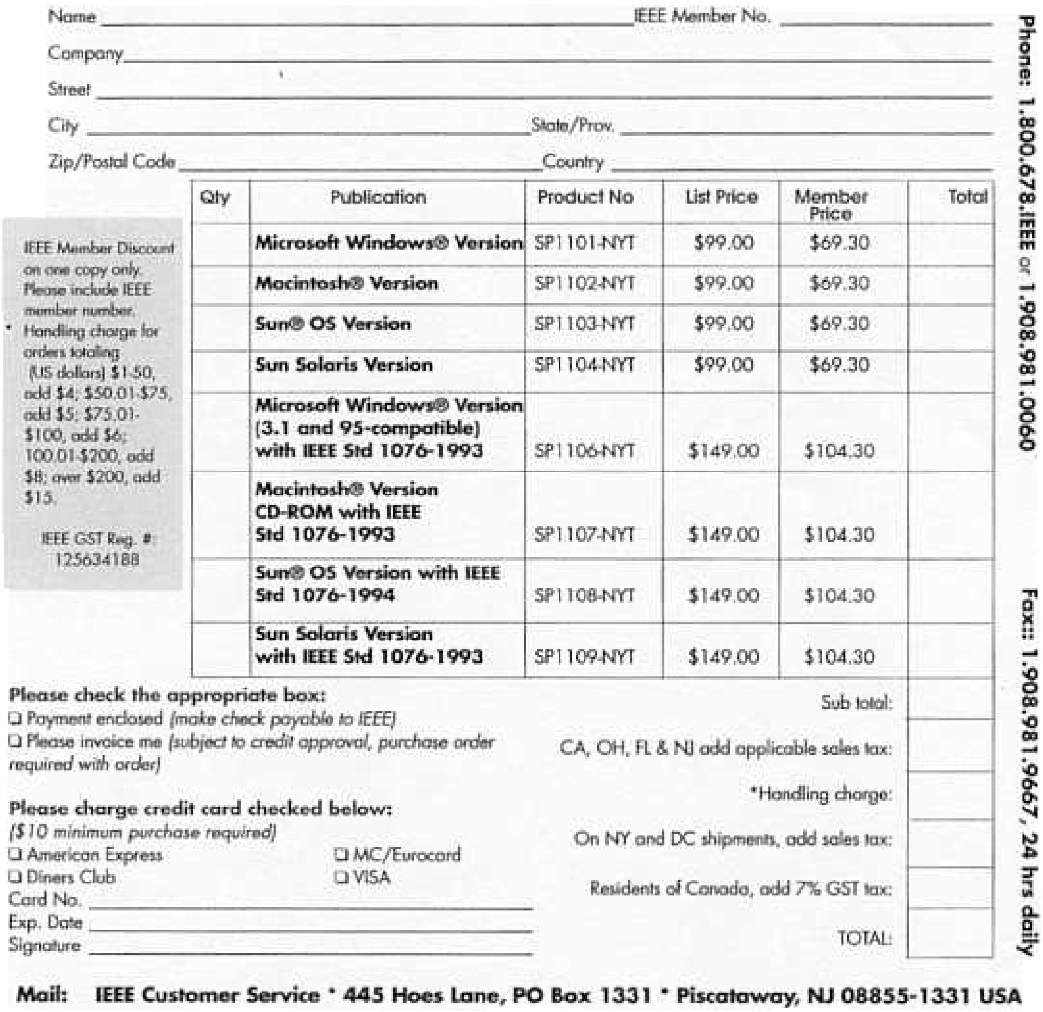
Module 1—Basic VHDL
Provides an introduction to the VHSIC Hardware Description Language and its fundamental concepts and describes the many advantages of using VHDL.

Module 2—Structural VHDL
Introduces the concepts and constructs of structural modeling using VHDL. It brings you to the point where you can write code using the concepts of structural design in VHDL. What's more, it addresses the use of VHDL for describing models in terms of component instantiations and interconnections.

Module 3—Behavioral VHDL
Describes features of the language that describe the behavior of components in response to signals. The VHDL constructs in this module focus on describing hardware utilizing software engineering practices and it concludes with a comprehensive example using the SDSP microprocessor.

Module 4—System Level VHDL
Covers a wide range of topics, focusing on VHDL constructs as applied to higher levels of design abstraction. This concentrates on the usefulness of VHDL at the system level while it presents examples in which VHDL is used to model the system at high levels of abstraction.

This product is licensed for single user purposes. Use for multiple-user environments will be available upon request. For more information, please e-mail stds.vhdlinfo@ieee.org or call 908. 562. 3804.



Name ______________________ IEEE Member No. ____________

Company ______________________

Street ______________________

City ______________________ State/Prov. ______________

Zip/Postal Code ______________________ Country ______________

Qty	Publication	Product No	List Price	Member Price	Total
	Microsoft Windows® Version	SP1101-NYT	$99.00	$69.30	
	Macintosh® Version	SP1102-NYT	$99.00	$69.30	
	Sun® OS Version	SP1103-NYT	$99.00	$69.30	
	Sun Solaris Version	SP1104-NYT	$99.00	$69.30	
	Microsoft Windows® Version (3.1 and 95-compatible) with IEEE Std 1076-1993	SP1106-NYT	$149.00	$104.30	
	Macintosh® Version CD-ROM with IEEE Std 1076-1993	SP1107-NYT	$149.00	$104.30	
	Sun® OS Version with IEEE Std 1076-1994	SP1108-NYT	$149.00	$104.30	
	Sun Solaris Version with IEEE Std 1076-1993	SP1109-NYT	$149.00	$104.30	

IEEE Member Discount on one copy only. Please include IEEE member number.

* Handling charge for orders totaling (US dollars) $1-50, add $4; $50.01-$75, add $5; $75.01-$100, add $6; 100.01-$200, add $8; over $200, add $15.

IEEE GST Reg. #: 125634188

Please check the appropriate box:

□ Payment enclosed *(make check payable to IEEE)*

□ Please invoice me *(subject to credit approval, purchase order required with order)*

Please charge credit card checked below:

($10 minimum purchase required)

□ American Express □ MC/Eurocard

□ Diners Club □ VISA

Card No. ______________________

Exp. Date ______________________

Signature ______________________

Sub total:	
CA, OH, FL & NJ add applicable sales tax:	
*Handling charge:	
On NY and DC shipments, add sales tax:	
Residents of Canada, add 7% GST tax:	
TOTAL:	

Phone: 1.800.678.IEEE or **1.908.981.0060**

Fax:: 1.908.981.9667, 24 hrs daily

Mail: IEEE Customer Service * 445 Hoes Lane, PO Box 1331 * Piscataway, NJ 08855-1331 USA

Institute of Electrical and Electronics Engineers, Inc.
IEEE Standards Press
445 Hoes Lane, PO Box 1331
Piscataway, NJ 08855-1331 USA
(908) 562-3829
http://stdsbbs.ieee.org/

VHDL Interactive Tutorial—Your Complete VHDL Training Tool

Nov. '96

APPENDIX E - *RASSP DIGESTS* (external)

Vol. 1, No. 1, 4th Qtr. 1994

RASSP Digest

RASSP - Rapid Prototyping of Application Specific Signal Processors

In This Issue

Welcome to the inaugural edition of *The RASSP Digest*, the quarterly newsletter for the U.S. Department of Defense's RASSP Program! The primary aim of the newsletter is to chronicle RASSP-related activities and to inform the general rapid systems prototyping community of developments and new results developed by the RASSP Program. Each issue will include notices of upcoming RASSP related activities and focus on one of the topical themes that impact rapid systems development. These typically include, but are not restricted to, an Executive Outlook section that presents the views and comments from a programmatic point of view and a Prime Development section that presents developments from the Lockheed Sanders and Martin Marietta programs relevant to the newsletter's current topical theme. In addition, a Technology Base section presents one or more articles highlighting the advanced technology being developed within the universities and companies of the RASSP Technology Base. The Benchmark section will provide an update on the results and issues of the RASSP benchmarking activity. Lastly, there will be an Editorial section that discusses recent viewpoints and offers technical articles on various issues in the area of rapid systems prototyping. In this issue, the editorial presents a high-level comparison of the RASSP approach vis-'a-vis current practice approach.

If you have ideas for topical themes to be addressed in future issues of the RASSP Digest, please contact the editors at the address below.

Dr. Anthony J. Gadient
Email: gadient@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

Dr. Vijay K. Madisetti
Email: vkm@ee.gatech.edu
Georgia Tech
Sch. of Elec. & Computer Eng.
Atlanta, GA 30332-0250

RASSP After One Year

Mark Richards, ARPA (ESTO), RASSP Program Manager

The Rapid Prototyping of Application Specific Signal Processors (RASSP) program is a major ARPA/Tri-Service initiative to reinvent the design process for embedded digital signal processors. Our goal is to improve the time it takes to go from concept to fielded prototype on both new designs and upgrades by a factor of four, with similar improvements in life cycle cost and supportability.

RASSP is aimed at the whole system design process, from specification to manufacture. The program emphasizes complex digital systems at the board and multi-board levels. Consequently, it spans heterogeneous systems involving mixes of standard and custom hardware, field programmable devices, and software on programmable processors.

At this writing, the RASSP Program is one year into the four year span of the primary development programs which are the core of the program. What has been accomplished in the first year?

Programmatically, the ARPA/Tri-Service team has finished the major undertaking of building the RASSP team and ramping the program up to full speed. RASSP now involves some twenty-five contracts in four general areas. Lockheed Sanders (Nashua, NH) and the Martin Marietta Advanced Technology Laboratory (Moorestown, NJ) each are developing and demonstrating complete RASSP design systems. They are supported by a technology base development effort involving a large number of universities, not-for-profit, and commercial firms performing research and development in digital signal processing and electronic design automation. RASSP also includes innovative Education and Facilitation (E&F) and benchmarking efforts, both new experiments in ARPA programs. Current design metrics emphasize evaluation of point tools. The RASSP benchmarking effort will develop system-level benchmark design problems and process metrics, and share RASSP design experience with the larger community. The E&F will ensure that RASSP design concepts and experience are widely disseminated to the larger end user, commercial EDA, and educational communities.

Technically, RASSP efforts are focused in three general areas: design process methodology, digital signal processor architecture, and electronic design infrastructure which includes EDA tools, libraries, and enterprise integration capabilities. While progress has been made in all three, I would like to concentrate on the first and third areas.

A good design system should be driven by methodology, tempered by available tools and other infrastructure. The RASSP ProgramRASSP Program is emphasizing development of a concurrent systems engineering methodology. Both Lockheed and Martin have defined first versions of a RASSP design process that includes a highly integrated functional design process (meaning the process of translating requirements into functional specifications and constraints, and then mapping those specifications into a hardware/software architecture optimized under the constraints), augmented with concurrent engineering capabilities such as early cost estimation, producibility assessment and so forth. A number of successes have been achieved in linking tools for DSP algorithm development to tools for system simulation and hardware/software codesign. The RASSP Program is also exploring the limits of VHDL as a language for system representation across many levels of abstraction, from executable specifications to behavioral and RTL-level designs suitable for synthesis, to gate -level design documentation. This effort has made substantial progress and is also illuminating many practical problems, from simulation speed to library population. The latter point brings us to the area of design infrastructure.

The RASSP Program has invested in its first year in a number of efforts to help populate VHDL part libraries, ranging from direct development of models to efforts to commercialize model generation tools. One of the most visible signs of our early progress will be the announcement in 1995 of a number of new or accelerated EDA products and enhancements directly attributable to these RASSP efforts.

RASSP faces a number of near term challenges. Before a second year elapses, the developers must integrate the first version of a comprehensive design system. The first benchmarking results will become available, and are certain to include many lessons about what is and isn't working in the RASSP methodology. Virtual prototypes of each developer's demonstration projects will be completed, providing another test of the early methodology and tools. Finally, we must also begin to make progress in our efforts to explore DSP architectural structures which inherently support rapid design.

While the RASSP Program is now up to full speed, opportunities still exist to work with us to meet the goal of dramatically improving embedded DSP design. The ARPA/Tri-Service team remains interested in partnering with end users on additional demonstration projects. And of course, we are always interested in hearing of good new technical ideas applicable to the RASSP Program.

The RASSP Education and Facilitation team is there to help you learn more about RASSP. This newsletter contains a great deal of information about all aspects of the program. To learn still more, I encourage you to log into the RASSP World Wide Web server at **http://rassp.scra.org/**, or contact the RASSP E&F team directly. And be sure to join us for the Second Annual RASSP Conference next Summer!

RASSP Education and Facilitation

Jack Corley
SCRA

The RASSP (Rapid Prototyping of Application Specific Signal Processors) Education and Facilitation (RASSP E&F) program is developing the innovative education and facilitation system needed to make RASSP technology widely used.

The RASSP E&F team has three primary objectives: 1) to transfer the RASSP knowledge and technology into use in defense and commercial industry, 2) to transfer the RASSP knowledge and technology into university curricula, and 3) to facilitate the continuous improvement of RASSP through rapid feedback.

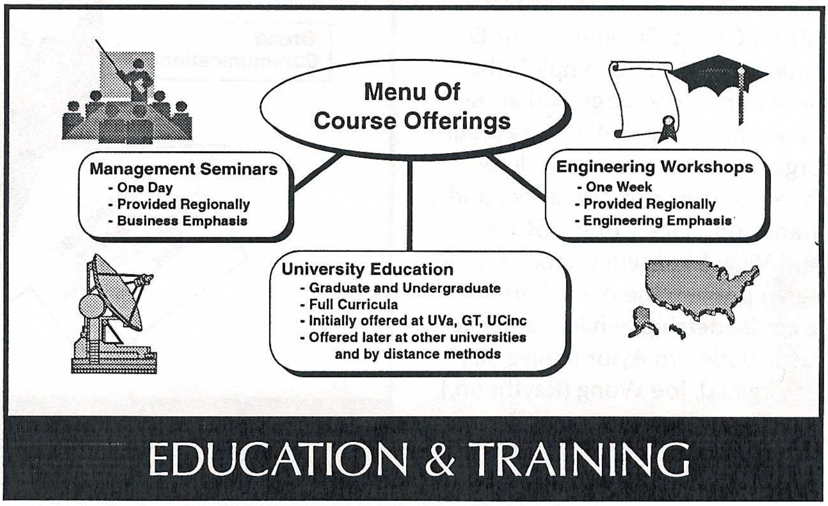
To accomplish these objectives, the RASSP E&F program is developing university and continuing education; a single point source for all RASSP information; and facilitating technology transfer to industry and academia. To stay abreast of RASSP technology development and provide user input to help steer the development, RASSP E&F maintains a strong interface with all other RASSP programs.

RASSP is developing the methods, tools, and DSP architecture needed for a paradigm shift in the way systems are designed, verified, and upgraded. That paradigm shift is needed to reach the RASSP objective of a four-fold decrease in design time with associated improvements in quality and upgrade potential. However, even if the RASSP technology were fully available today, there are very few educators or engineers who are familiar with the key facets of that technology.

Other components of the RASSP initiative are eliminating design tool, DSP architecture and methodology gaps. The RASSP E&F program must address the scarcity of trained industrial technical staff and educators. Unless these individuals are trained in the effective use of RASSP concepts, RASSP objectives can not be met. Education and Facilitation are vital if commercial and defense companies are to realize the RASSP benefits. RASSP E&F must both increase the supply of, and accelerate the demand for skilled RASSP technologists.

Our RASSP E&F approach recognizes the large diverse audience of management, engineering, and university people that must be reached and their differing needs and objectives. Different techniques will be used to reach that diverse community, emphasizing the portions of the overall RASSP message relevant to the particular target audience.

The RASSP E&F program will develop a RASSP Education System (RES), demonstrate the RES effectiveness, and ensure its long-term availability as the innovative education and training needed to make RASSP technology widely used. RES will be developed and delivered in both academic and industrial settings. This education will act to increase both the supply of, and the demand for RASSP qualified engineers.



The RASSP E&F goal is to collect, consolidate, and disseminate RASSP information and technology. RASSP E&F will educate the user and facilitate proliferation of RASSP into industrial practice and RES into university curricula to change the way embedded system design is performed and taught. RASSP E&F will also provide users with a mechanism to feedback their perspective to RASSP developers, ensuring that RASSP is continuously improving its focus on key user issues.

The RASSP E&F team is working with industry and academia to identify the issues that must be addressed for successful change. The entire RASSP E&F offering is tailored to meet those needs. Careful attention to overcoming barriers is being coupled with innovative learning and information dissemination techniques to ensure the target audience is receptive and RASSP technology widely used. RASSP E&F will continually monitor and improve the effectiveness of the RES approach, delivery mechanisms, and information content.

The RASSP E&F team consists of leading university professors, technical managers, electrical engineers and computer scientists from SCRA, Georgia Tech, University of Virginia, University of Cincinnati, Raytheon, Merkel and Mears Group, EIT and Arthur D. Little (ADL). To accomplish the necessary technology and knowledge transfer, RASSP E&F has four organizational segments: Interface, Education, Information, and Transition. Jack Corley (SCRA) and Vijay Madisetti (Georgia Tech) provide the overall program leadership, while team leaders include Jim Aylor (University of Virginia), Joe Wong (Raytheon,) Hal Carter (University of Cincinnati,) and Anthony Gadient (SCRA).

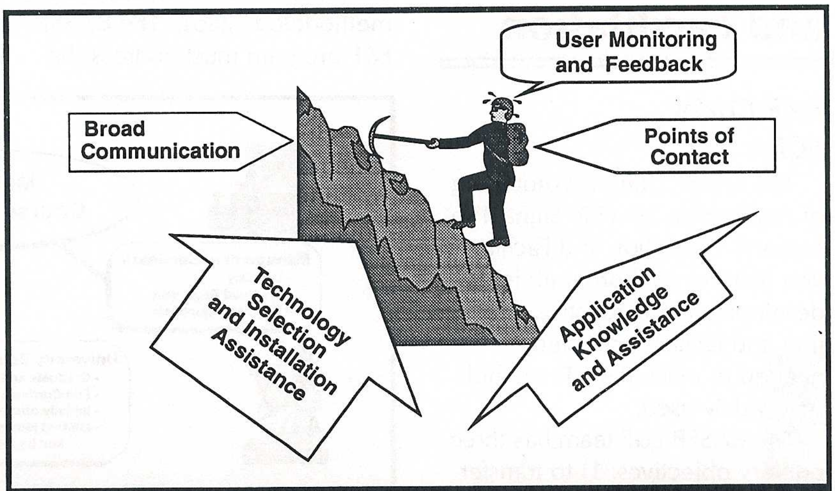
The RASSP E&F team is using a spiral, continuous improvement approach to deliver education and facilitation. This spiral consists of four steps: 1) define/refine objectives, 2) define/refine infrastructure and requirements, 3) develop/refine mechanisms and metrics, and 4) deliver and evaluate effectiveness.

An innovative approach to modular courseware is being used by the Education segment. This modular approach provides a simple, cost-effective way to continually improve the material, use the same material for multiple purposes, and stay abreast of the RASSP technology improvements. Each module will contain three components: 1) theory and fundamentals, 2) examples and metrics, and 3) detailed RASSP systems design examples linked to RASSP tools and methodologies. This unique approach simplifies course creation, maintenance, and transfer to educational institutions outside the initial RASSP E&F team.

To reach the geographically dispersed audience, RASSP E&F is making maximum use of computing and communications facilities for innovative instruction delivery (multimedia, Internet, video-based instruction, etc.) State-of-the-art Internet communication tools will provide a logical single-point interface to on-line education materials, information and services from RASSP E&F, the other RASSP programs and EDA vendors. The same Internet tools will be used for team collaboration and to interface with other RASSP programs.

Through these distributed education and facilitation mechanisms, RASSP E&F is providing the apparatus needed to make RASSP technology broadly available. The overall effect will blur the boundary between information, education, and use of RASSP services. Users will skip introductory and tutorial material as they become expert, with on-line reference manuals, interactive help and refresher training only a click away. Users will progress effortlessly from novice to expert. The end-result should make RASSP technology ubiquitous, making the RASSP four-fold improvement goals broadly realized.

Your comments and suggestions are vital to the continuing improvement of RASSP and


User Monitoring and Feedback
Broad Communication
Points of Contact
Technology Selection and Installation Assistance
Application Knowledge and Assistance

RASSP E&F FACILITATION

RASSP E&F. Contact information to remember includes:

Phone: 803-760-3376
Email: info@rassp.scra.org
ftp: ftp.rassp.scra.org
World Wide Web
(Mosaic and Lynx viewers)
http://rassp.scra.org
We need your help for RASSP to meet its goal.

Conceptual Prototype Demonstrates RASSP's Future

Harley Stein
Martin Marietta

Martin Marietta's RASSP conceptual prototype is a forward-looking view of the future RASSP workstation. It shows the steps the Martin Marietta team is focusing its efforts on to achieve the RASSP 4X design time improvement.

Common Desktop Environment Provides Five Keys

The conceptual prototype environment is based on the Common Open Software Environment (COSE) consortia's Common Desktop Environment (CDE). Martin Marietta developed several skeleton applications using CDE interface builders. These skeletons simulate functionality that is not available in "real" applications. "Along with the skeletons, we've augmented the conceptual prototype with screen shots of actual applications," said Andrew Schwalb, developer of the Martin Marietta conceptual prototype. "This gives it a realistic look and feel."

The screen shots help show the functionality provided by the different applications through the entire RASSP framework. The prototype will evolve throughout the program to encompass added functionality as the tools mature.

Martin Marietta selected CDE as its standard interface specification for RASSP for five key reasons:

-Unprecedented interoperability
-Efficient user interaction
-Transparent network support
-Virtual workstations
-Shared groupware applications

CDE is being co-developed and embraced as the standard interface application for nearly all major workstation vendors, including Sun Microsystems, Hewlett-Packard, IBM, Novelle, and recently, DEC. This specification will provide a common look and feel, and ensures compatibility for all Martin Marietta's RASSP applications. The RASSP design philosophy is to use the full capabilities of CDE without incurring the cost of developing a totally new environment.

Workspaces Combine Common Data

One of the primary internal features is the virtual workstation, which allows users to segregate data related to a project into organized, easy-to-access categories. These categories, or workspaces, combine common data into a single environment. The workspaces currently defined for the RASSP enterprise system are:

-Project workspace
-Workflow workspace
-Product/library data workspace
-Network/Interfaces workspace
-Design tools workspace
-General user workspace

Martin Marrieta plans to refine the workspaces as data organization is refined during the course of the RASSP Program.

Interoperability Demonstrated in Key Areas

Interopertability is the key to a successful RASSP Program. Martin Marietta's conceptual prototype shows some of the key areas that its team mates are developing, including project information workspaces, the link between the workflow manager and several tools, the use of desktop video conferencing tools, and networking.

The project information workspace example demonstrates how a project management tool gets updated status information provided by the workflow management tool. This link enables RASSP users to see status on a Gannt chart, reference a step within a workflow, and find any ties to a requirement. "All this information is available with a few keystrokes," said Schwalb. "The burden of data management is on the RASSP design framework, not on the users."

The workflow manager integrates the design process with tool launch, design review/authorization, and data management capabilities. The tie between the workflow manager and the tools provide users with transparent data access following a workflow. The demonstration shows the power of abstracting users from the underlying data management details using several examples.

The desktop video conferencing tools enable users to interact remotely with engineers, project managers or customers without leaving the workstation. The prototype shows a design engineer grappling with an application problem. The engineer immediately connects with support staff who talk the engineer through the problem, and who can also take control of the application and show the engineer how to solve the problem.

The network example shows an engineer using the system to select a location to manufacture a product. Through the network the engineer can assess the manufacturing center's capabilities, send a preliminary design to the center, and receive feedback regarding the manufacturability of the product.

The Martin Marietta conceptual prototype was unveiled at the first Annual RASSP Conference. It is available for demonstration at Martin Marietta Laboratories/ Moorestown in Moorestown, New Jersey. The evolving conceptual prototype was demonstrated at the 1994 GOMAC and will be demonstrated at 1995 DAC.

Martin Marietta RASSP Design Center Enables Design Environment Implementation

Lynn Kline
Martin Marietta

Martin Marietta's RASSP design center provides a facility for the team to implement the integrated RASSP design environment. The RASSP design environment is important because it enables a significant portion of the 4X improvement in development schedule and 4X reduction in life-cycle cost.

The RASSP design center at Martin Marietta Laboratories in Moorestown has a 30-Gigabyte data server and 10 Sun Microsystems' Sparc 10 workstations dedicated to RASSP. The team plans annual hardware enhancements using Martin Marietta capital. These workstations are networked into the Moorestown Laboratories' resources. Martin Marietta plans to connect to the outside world in early 1995 using ElNet from MCC to provide flexible and secure on-demand connectivity.

Martin Marietta's design environment implementation team leveraged the heritage of its Engineering Process Improvement (EPI) program to combine an already integrated set of CAD tools with additional DSP analysis tools to implement its Baseline 0 design environment. Martin Marietta has now defined a set of 46 tools from 26 vendors. Intergraph will provide a framework to integrate all tools and automate process and workflow control. These tools are fully described in Martin Marietta's "CAD System Description" document. The tools are organized according to their use within the RASSP design methodology.

The tools for the systems, architecture, and detailed design (hardware and software) areas are summarized in the following paragraphs.

System Design tools support early development of system partitioning, test, reliability, and maintenance concepts.

RTM by Marconi System Technology enables life-cycle requirement traceability

PRICE S/M/H/HL by Martin Marietta for computer-aided parametric cost estimating enables life-cycle cost analysis throughout the design process

RAM/ILS by Management Sciences Inc. enables feedback on reliability, availability, maintainability, and integrated logistics support during early trade-off analyses

RDD-100 by Ascent Logic enables system definition, functional analysis, function allocation, interface design, scenario development, and thread analysis

TPS by Interleaf provides a complete document publishing tool

RRDM by Aspect will provide access to reuse data and libraries

Architecture Design tools help analyze architecture and hardware/software codesign in a variety of ways: functional analysis, trade-off studies for architecture selection, partitioning/mapping, and architecture verification.

NetSyn by JRS Research Laboratories enables multiprocessor design analysis and synthesis to support architectural trade-offs

SPW by Alta Group of Cadence enables interactive design, simulation, and implementation of digital signal processing and communication systems

BONeS by Alta Group of Cadence performs detailed, discrete architectural simulation and is used to obtain high-fidelity performance metrics early in the system design

MatLab by Mathworks provides advanced image processing functionality and numeric computation

Ptolemy by BDTI/UC Berkeley supports multi-domain analysis of complex systems

GEDAE by Martin Marietta provides a software front-end for hardware testbeds that enables graph-based programming and front-end analysis of multiprocessor trade-offs

ADEPT by University of Virginia provides a unified VHDL environment to support hardware/software codesign and trade-offs

Hardware Design tools support seamless coupling from the higher level architectural requirements, hardware/software codesign, and behavioral tools down to the functional and detailed-level hardware design processes.

Design Architect by Mentor Graphics captures designs at the architectural, logic, and circuit levels for top-down design

QuickVHDL by Mentor Graphics creates, debugs, and simulates VHDL models

QuickPath by Mentor Graphics provides static path analysis

QuickFault by Mentor Graphics provides deterministic fault-simulation

MCM Station by Mentor Graphics provides layout, thermal analysis, and signal integrity analysis of MCMs or PWBs

DesignVision by Vista graphically represents behavior for modeling VHDL, viewing simulation results, and documentation

SimMatrix by Precedence provides a simulation backplane to support interactive co-simulation

FIDELITY by Omniview allows designers to rapidly synthesize and evaluate alternative hardware architectures

VPS by Quickturn Design Systems enables hardware emulations

SmartModels by LMG group of Synopsys provides full and bus functional simulation libraries for a large number of COTS parts

Design Compiler by Synopsys enables high-level design synthesis of ASICS

C-MDE from LSILogic enables ASIC development

FPGA Foundary from Neocad supports FPGA development

Lasar by Teradyne provides dynamic min/max timing path analysis

Victory by Teradyne provides test analysis

Software Design tools support library development, detailed design, and source code development. Martin Marietta will be adding more software tools soon.

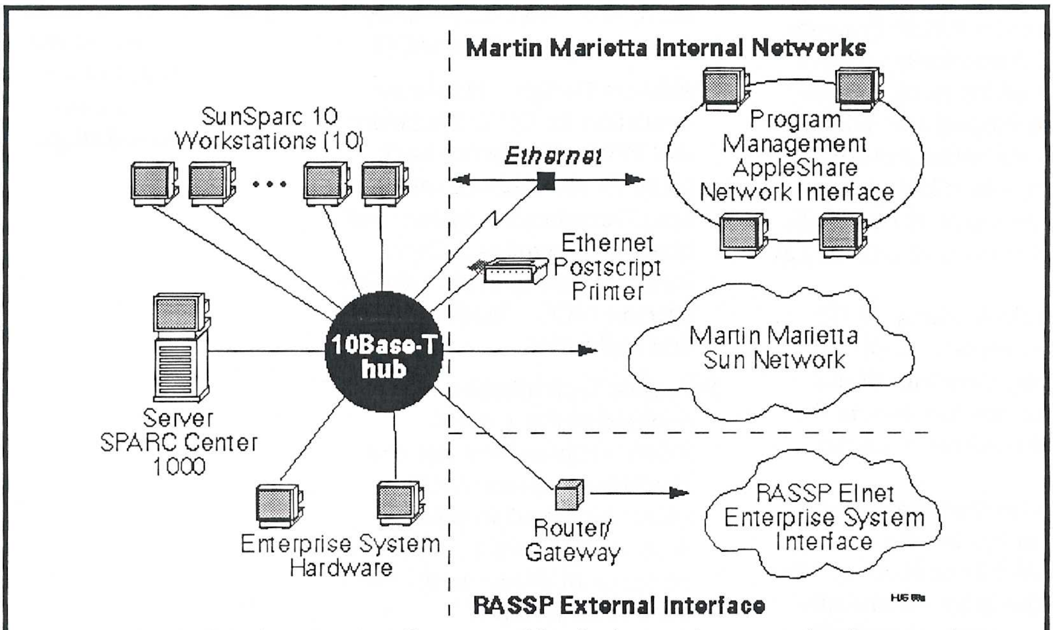
Teamwork by Cadre Technologies provides structured design analysis and documentation in support of software development

Sun ADA by Sun Microsystems supports HOL source code development in the workstation environment; target-specific HOL compilers are being installed to support emerging DSP chips

CDEM will be provided by AT&T to support distributed software debugging capabilities for multiprocessor systems

PIE and **TIBBIT** by University of Oregon will provide a performance analysis and a binary-to-binary software translation capabilities

GrTT and µPIDgen by Management Communications and Control Inc. will provide autocode generation and run-time system for graph execution control



The Martin Marietta Design Center enables design environment implementation

For more information regarding the RASSP Design Center or its tools, contact Lynn Kline at 609-866-7191.

The Martin Marietta RASSP Team Demonstrates and Presents Rapid Prototyping Concepts at the First Annual Conference

Stephen O'Neill
Martin Marietta

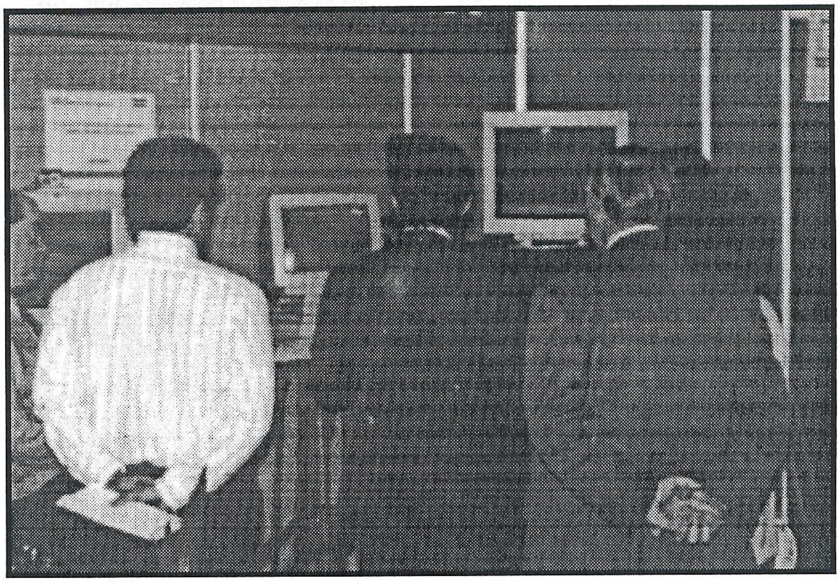
The Martin Marietta RASSP team demonstrated 16 rapid prototyping developments at the first Annual Conference at the Hyatt Regency in Crystal City, Virginia August 15-18.

The ARPA/Tri-Service-sponsored four-day conference was part of the government's mission to publicize the RASSP Program, which will dramatically improve the way in which signal processors are developed and fielded. Eventually, the rapid prototyping technology, which has concurrent engineering at its core, will be applied to digital processing at large.

The Martin Marietta RASSP team members also presented seven papers covering various parts of the development to the nearly 400 conference participants.

The Martin Marietta exhibit consisted of five booths. In one, it demonstrated its concept of operations. In the other four its subcontractors demonstrated various items of the program's enterprise system, architecture, hardware design, and pervasive technologies:

Enterprise System - Enterprise integration: Intergraph and Mentor - Library management integration: Aspect and Mentor - Manufacturing interfaces: SCRA - Electronic networking:



MCC Architecture - Multiprocessor network synthesis: JRS - Simulation interoperability: BDTI, Alta, and U.C. Berkeley - Autocode generation: MCCI

Hardware Design - Hardware emulation for DICE: Quickturn and TRW - Simulation backplane for architecture verification: Precedence - System and board-level synthesis: Omniview - Multichip system design advisor: MCC - Model generation tools: LMG

Pervasive Technologies - Object-oriented VHDL extension: VISTA - Hierarchical test and economics advisor: MCC - Hierarchical built-in self-test design: LV Software - Parametric cost modeling: Martin Marietta PRICE Systems.

A large percentage of the Martin Marietta team members are EDA industry leaders and have committed to release early versions of their developments. One of the many representatives from the signal processing and CAD industry commented that "the Martin Marietta team had demonstrated a world class capability."

Companies interested in becoming beta sites for the RASSP methodology and design system should contact:

Jim Saultz
(609) 866-6402
via e-mail
jsaultz@atl.ge.com.

Introduction to the Lockheed Sanders RASSP Team

Cory Myers
Lockheed Sanders

In order to meet the goals of RASSP, Lockheed Sanders teamed with Motorola, Hughes Aerospace, and ISX. The technical work on the program is split nearly equally between Lockheed, Motorola and Hughes while ISX has a small, but significant role. Lockheed Sanders brings demonstrated expertise in rapid development and signal processing. Hughes brings extensive signal processing and military system development. Motorola brings recognized expertise in process improvement and in communications. ISX demonstrates skill in the management of large consortia.

Because of the success which Lockheed and the US Air Force have had on the F-22 program with the use of Integrated Product Teams, we chose to use a similar management technique on RASSP. Our program consists of four Integrated Process and Product Development Teams. These four teams include Systems, Design Environment, Demonstration, and Proliferation (Figure 1). All four companies have significant roles in all four teams.

Lockheed leads the Systems team as well as being the prime contractor on the program. The Systems team provides the methods and architecture that will support rapid development. These include the model year concept, the extensive use of virtual prototypes, the ability to "plug and play" hardware and software components, and extensive use of reuse libraries.

Motorola has the lead responsibility on the RASSP Design Environment (RDE). The RDE provides a flexible environment for the management of the development process, including work flow management, configuration management, metric collections, and communications. The RDE (as well as other services, including "pay-per-view" tool rental) will be provided commercially through the Electronic Information Corporation.

Hughes leads the Demonstration team. (Our demonstration is an IRST upgrade to the Navy's F-14). The demonstration team proves our methodology and environment while providing rapid feedback to the developers as to potential improvements.

ISX has the lead responsibility for the Proliferation team. The Proliferation team distributes the RASSP process and the RDE to beta sites to further demonstrate the process and to provide additional feedback to the developers.

The remainder of this article provides the reader with a better understanding of our RDE, our demonstration efforts, and our proliferation work. The RASSP Approach section gives a description of the organization of our system and how it will enhance the rapid development of signal processing systems. The Demonstration section gives a detailed

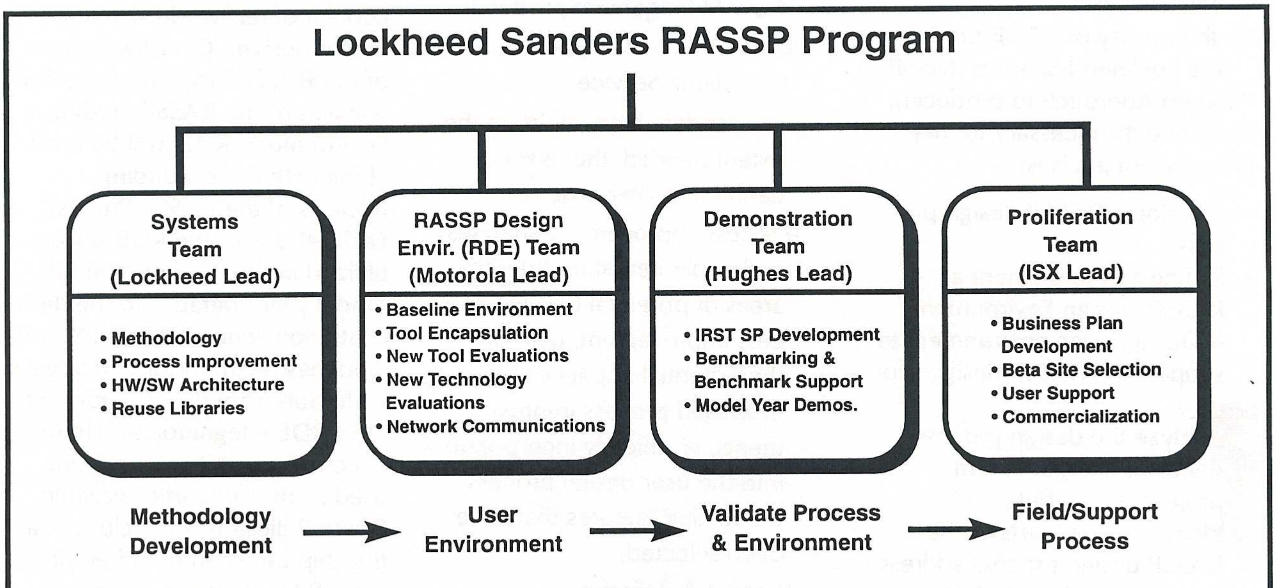


Figure 1. Organization of the Lockheed Sanders RASSP Team

description of the problem that we are undertaking and the methods that we are using. The Proliferation section describes our approach to beta sites and gives a glimpse at the future of the Electronic Information Corporation.

The Lockheed Sanders RASSP Approach

Ron Ireland
Motorola

Driving the Lockheed Sanders RASSP Program product is the program goal as stated by Program Manager Mark Richards:

> ***"Dramatically improve the process by which embedded digital signal processors are designed, manufactured, upgraded and supported."***

This is put into quantitative goals of improving by a factor of 4:

1) the total cycle time to produce a DSP product,
2) the cost of producing a DSP product, and
3) the quality of a DSP product

The Lockheed Sanders RASSP Program Approach to producing the products necessary to meet the program goals is:

- Develop a RASSP design process
- Define and implement a RASSP Design Environment (RDE), a design environment to support the RASSP design process
- Analyze the design process needs of a target design project (e.g., a beta site)
- Identify which parts of the RASSP design process address the target project design process needs
- Incorporate into the target project design process those selected RASSP design process features deemed to be needed (process mapping)
- Determine the project design environment improvements needed to support the new project process
- Provide to the project those RASSP Design Environment features which have been developed by the Lockheed Sanders RASSP team.

The Lockheed Sanders Program Product (those services and software products needed to support the approach described above) includes the following:

- A RASSP Design Process
- Deliverable Software (This software is developed by the Lockheed Sanders RASSP development team. It is not COTS software)
  - Integration and other Specialized Software (metrics collection, metrics reporting, user interface software to enhance user friendliness)
  -Customization Files
  -Process description for Work Flow Management products
  -Encapsulation wrappers
- Consulting Service
  -Process definition (assist, to the extent needed, the user in defining his/her process)
  -Process improvement analyses and implementation (identify areas of potential design process improvement, quantify the potential impact of the proposed process improvements, seamlessly incorporate into the user design process the RASSP features that have been selected)
  -Process Automation
  -Distributed Document Management automation
  -Design environment definition (Features needed to optimally support the user design process)
  -Customization of COTS software (Process instantiation using a COTS Work Flow Manager, Distributed document management using a COTS document management product plus selected RASSP provided features, Application Encapsulation)
  -Metrics definition, automatic collection and analysis

The Lockheed Sanders Demonstration Program

LeRoy Fisher
Hughes

The goal of the RASSP Demonstration effort is to provide validation of the RASSP Process and RASSP Design Environment in the context of real-world signal processor design. Over the course of the RASSP Program, three full releases of the RASSP Design Environment (RDE) will be used, along with corresponding releases of the RASSP Process. Each release of the RDE will be utilized in the development of model year upgrades to the demonstration vehicle. Model Year 0 work has been performed largely with tools from the RDE tool set. New RDE integration and infrastructure capabilities are being used as they become available. Figure 2 illustrates the close relationship between the three primary RASSP efforts - Process Development, Design Environment Development and Demonstration.

The objectives of the RASSP demonstrations are to:

1) Use an embedded signal processing system as a test case, spanning the development cycle from concept to specification, architecture analysis, design, manufacture and support, so that the entire RASSP process can be evaluated as it evolves during the contract.
2) Design the system using the RASSP model year concept - the ability to upgrade system design rapidly and often, incorporating the latest technology and incrementally upgrading the system throughout its life cycle.
3) Provide process metrics and lessons-learned for methodology and process refinement. Measure the progress toward reducing product development time by a factor of four.
4) Provide feedback on the usefulness of specific tools and the design environment.
5) Provide clear, convincing data that the RASSP methodology is practical and effective for complex design tasks.

By demonstrating the process and tools as they are being defined, the program becomes much more than an esoteric study. The demonstration helps focus and prioritize development. It also checks the usefulness of the tools early in the development cycle. Adequate design complexity ensures that the process and tools are viable in real-world examples. Because requirement definition is real, there is no opportunity to gloss over details, as might be possible for a simple laboratory demonstration. A tangible demonstration also provides an understandable scope of work for which metrics can be collected and improvements in the RASSP process explained. Metrics gathered across multiple years clearly show that the design process has improved.

Demonstration Vehicle

The demonstration vehicle for RASSP is an airborne infrared search and track (IRST) processing system using programmable processors. It employs a heterogeneous Multiple Instruction Multiple Data (MIMD) architecture using commercial off the shelf (COTS) processor chips, operating systems, and system software tools. The IRST processor was chosen as the preferred demonstration for several reasons:

1) **Scalability**. The IRST algorithms were available in a scalable manner for a coarse grain MIMD parallel processor. Coarse grain in this sense is a non-shared memory, message passing architecture. The available software design and initial partitioning provided a starting point for showing model year improvement and demonstrating reuse. By starting from this well-thought-out design, a more complex system demonstration can be achieved.
2) **Modularity**. The IRST algorithms and software are modular to allow RASSP process exploration to be performed at different levels of rigor for different functions. This modularity allows the scope of the demonstration to be adjusted to meet RASSP needs and schedule.
3) **Tools**. Some of the algorithms are described at the

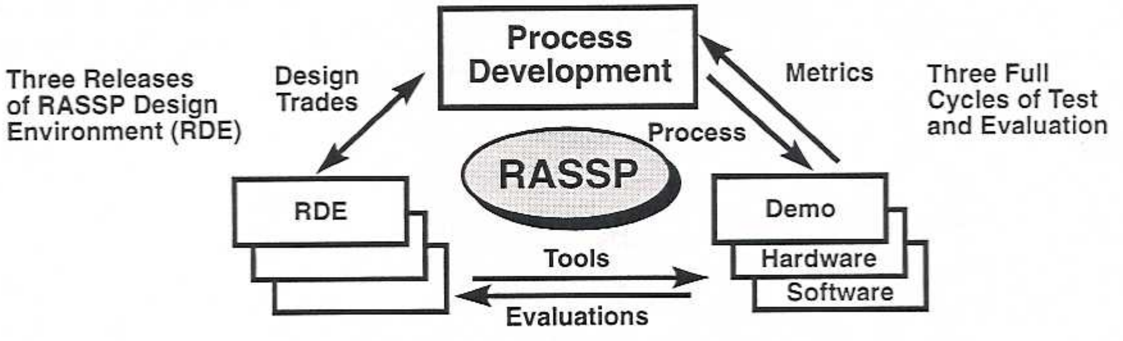


Figure 2. Relationship Between RASSP Activities

math level in Matlab,' one of the RASSP analysis tools, thus allowing a top-down exploration of hardware/ software partitioning and design.

4) **Ease of implementation.** Some of the existing algorithms easily lend themselves to hardware implementation (e.g., convolution and registration). Also, test imagery is available.

5) **Applicability.** IRST processing has received much recent interest as a means to meet airborne and ship defense requirements. Thus its demonstration should be of interest to multiple potential Government organizations. Ideally, these organizations can start with the RASSP demonstration databases, apply the RASSP process, and create their own IRST solution tailored to specific application requirements.

Demonstration Model Year 0 concludes with the construction of a complete IRST processing system. Algorithms and some C software were available at the beginning of the Model Year 0 effort. The architecture, interface cards, processor, and Ada software are the design elements for this first model year. The design uses standard buses and interfaces, such as VME, RS-170, and RS-422. The system will have about 20,000 lines of application code and about 24,000 lines of VHDL code.

Lockheed Sanders Beta Site Program

Ron Ireland
Motorola

The Lockheed Sanders Beta Site Program is an integral and critical part of the overall RASSP Program. This part of the program is where real users outside of the Lockheed Sanders Team deploy the Lockheed Sanders RASSP Product within their business. This part of the program is where we learn how well we really did in addressing the RASSP Program objectives.

The Lockheed Sanders philosophy is to involve the beta site projects early in the life of the program. Candidate projects are kept informed of the program direction and status. In turn, these candidate sites provide us with their input on our direction. Through this close working relationship, the beta sites will be provided first access to RASSP Products including:

- Process improvement recommendations
- Design environment recommendations needed to support process improvement
- Integration software produced by the Lockheed Sanders RASSP team
- Special purpose software produced by the Lockheed Sanders RASSP team.

As the RASSP products are deployed at the beta sites the Lockheed Sanders Beta Site Support Team will provide high levels of support and will continue to work with the user team members throughout the life of their project.

Over the next year the beta site program will provide the Lockheed Sanders RASSP Products to three different sets of users:

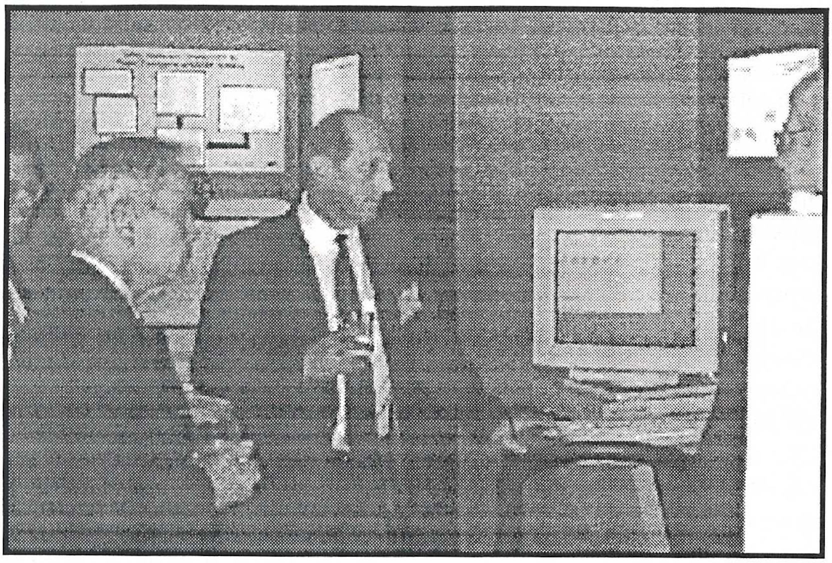
- Internal (to the Lockheed Sanders Team) projects - initiated in October 1994
- Selected government labs - initiated in January 1995
- First external projects initiated in 3Q95

The status of the beta site program can be described in terms of each of these sets of users:

- Internal beta sites have been identified; the definition of the packages that will go to these projects are being developed
- We are providing information to government lab projects who are interested in being external beta sites
- Dialog with several companies who have potential beta site projects is on-going

For additional information on the Lockheed Sanders Beta Site Program or to learn how you might become a beta site contact:

Ron Ireland, Manager
Lockheed Sanders RASSP Beta Site Program
Motorola-GSTG
8201 E. McDowell Road
Scottsdale, AZ 85252
Phone: 602-441-2348
email:Ron_Ireland@rassp.mot.com



RASSP Conference Success

Mark Hoffman
ISX

The Lockheed Sanders prime contractor team provided a booth area at the conference. In this area, a RASSP videotape was continuously run providing an introduction to the Lockheed Sanders team and its approach to RASSP. Also provided in the booth area for Sanders was a set of demonstrations illustrating technologies and methodologies being developed and employed by that team to address the RASSP goals of 4X improvements in the areas of cost, time, and quality. These demonstrations included:

The RASSP Design Environment (RDE) Release 0.1:

The product of the first year of RASSP effort, this prototype was the first of four stepping stones toward a RASSP environment and concentrated on document/data control and access and support for bug reporting.

RDE 1.0 Work Flow Manager:

This demonstration illustrated many of the features of the RASSP 1.0 work flow manager scheduled for release in June '95. The work flow manager was loaded with the current RASSP process definition to illustrate the contributions of reuse and concurrent engineering to the goals of 4X.

RASSP Visionary Demonstration:

This demonstration is a Macintosh based, multimedia program illustrating the "vision" of a future RASSP environment. This view was constructed by the team to help focus their own views on what a RASSP environment should provide.

RASSP Virtual Prototype:

This demonstration of the Virtual Prototype was shown by members of the Demonstration Team. The VHDL demonstration showed the MCV9/ISA model executing an application program example and compared the results with a similar application program executing directly on the workstation.

Multithreading Multiprocessors Demonstration:

The multithreaded, multiprocessor demonstration illustrated the use of a multi-processor workstation as a VHDL simulation accelerator. Simulation acceleration is essential as VHDL model complexity increases.

WWW Mosaic demonstration:

The Lockheed Sanders RASSP team has constructed a detailed World-Wide-Web Mosaic home page providing information on the team, its organization, goals, plans, and status. Most of this web is available to the public via URL "http://rassp.sanders.com."

Video Teleconferencing Demonstration:

The video teleconferencing demonstration illustrated the use of freeware video and screen-sharing tools used by the team in order to aid in the development of the RASSP environment itself. In this way, the Lockheed Sanders team is using components of their own RASSP environment and methodology in its own development.

Several of the Technology Base BAA contractors were also located in the Lockheed Sanders booth area including:

- University of California, Berkeley
- MIT/Boston University
- Research Triangle Institute
- GA SEER
- Georgia Tech
- CFI (CAD Framework Initiative).

Points of Contact

Additional information about the Lockheed Sanders RASSP Team and their plans and status can be obtained from several sources. A Mosaic World-Wide-Web home page is available through URL **http://rassp.sanders.com.**

The Lockheed Sanders RASSP Team also supports an automated fax-back service at 1-800-99RASSP. This number may be used to request additional specific information through an operator. Electronic mail requests for other RASSP information can be made through rassp-info@rassp.sanders.com.

VHDL Models

Hal Carter
University of Cincinnati

In this month's Tech Base column we will present the efforts taking place to create VHDL models for RASSP. Future columns will present Tech Base efforts in other areas such as simulation, synthesis, and translation tools; analysis algorithms and tools; and enterprise integration.

The information below generally describes the nature of the models being created and how they fit into the RASSP Program. A later issue of *The RASSP Digest* will provide more detailed information about each model and how they can be obtained from the RASSP repository.

VHDL Hybrid Models

Fred Rose, Honeywell
Jim Aylor, Univ. of Virginia

Hybrid models will allow the RASSP designer to seamlessly use multiple abstraction levels within the same simulation. This program will develop models and utilities to support hybrid modeling. VHDL is being used to model designs from early algorithmic and performance levels to behavior to detailed gate design. The primary advantage for using VHDL throughout the design cycle is keeping the design in the same representation throughout. The expressive power of VHDL is well suited to this task. However, unless techniques which allow seamless integration of these multiple levels are developed, this great advantage of VHDL will be lost. Individual tools and design representations will continue to be used, maintaining the current disarray and confusion in the industry for system level design. While the tailored RASSP VHDL DID calls out multiple VHDL levels, no technique exists to integrate these multiple models. In many cases, unrelated tools will be used and VHDL will merely be an output format ,almost an afterthought. Unless a clean, straightforward technique is developed, this will be true on the RASSP Program.

The complexity of model integration comes from different information content and format for the different model levels. Performance models use a token (VHDL record structure) as a signal which contains no detailed data information. Behavioral models use integer and real signal structures which contain detailed data information. Register transfer level models use bit level signal structures, such as the IEEE 1164 9-state logic system, which also contain detailed data information. Each of these levels also handle timing differently.

There are multiple reasons to mix levels. One may not have detailed design information for a particular block and may be forced to abstract the behavior. One may want to develop a detailed design for another block. Simulation performance may not be satisfactory with detailed models at all levels. Detailed models are used to verify timing assumptions made earlier. Software analysis may be desired but a detailed hardware processor model may not be available or may be too slow. These are all commonplace design occurrences.

Performance Modeling Workbench

Charles Buenzli, Omniview
Fred Rose, Honeywell

The Performance Modeling Workbench (PMW) will be a fully integrated VHDL environment for system-level hardware/software codesign and performance analysis. PMW will minimize the "up-front" model generation penalty through a comprehensive library of parameterized VHDL hardware and software models that allow the system designer to quickly generate and evaluate alternate architectures early in the system design process. A powerful, comprehensive results analysis package will allow the designer to compare and verify performance at each stage with previous results and the system requirements. Since all models are in VHDL, each stage of the design process is documented in a standardized, consistent, and verifiable form. The PMW will be based on the existing Honeywell performance modeling library.

VHDL Modeling of Architectural Building Blocks

Joanne Degroat, Ohio State University

The objective of this effort is the creation of a library of synthesizable (and simulatable, of course) common building block components that can be used to quickly design and implement a DSP or an SIC for use with a DSP. The library and associated documentation will provide a "data book" of reusable components with information on the use of the model for simulation and synthesis. Models with appropriate variations will be generated under the effort for fixed point integer units including adders, multipliers, linear shifters, and barrel shift/rotate units; and floating point units including single, double, and extended precision floating point adders, floating point multipliers, and transcendental function units.

VHDL Modeling of Cypress Semiconductor Standard Parts

Scott Calhoun, Mississippi State University & Cypress Semiconductor.

The objective of this effort is to develop VHDL models of selected Cypress Semiconductor standard integrated circuits. Mississippi State and Cypress have entered an agreement where by Cypress will provide timing information necessary to create VHDL models of high quality and accuracy to be released as part of the RASSP Program.

Several critical part types will be modeled to develop a baseline VHDL model library of standard parts. Advanced PLDs, FPGA, FIFO, and Dual Port memories are the part types under investigation. A meeting is scheduled between Cypress and MSU for later in the Fall to determine the first part to be modeled and the modeling support information which will be provided to MSU by Cypress.

MSU will also be releasing two VHDL modeling support tools as part of the RASSP contract. The first is TestView which is an automatic VHDL testbench synthesizer. TestView allows users to develop custom VHDL testbenches for the model under test (MUT) which are portable to any IEEE-1076 simulator. ConfigView provides a graphical user interface to VHDL configurations. ConfigView allows large configurations to be easily edited to customize simulation executions. Each of these programs will be released as MOTIF and Sun Microsystems COSI applications. Beta release for TestView is scheduled for 2nd quarter 1995. ConfigView Beta release will follow in 3rd quarter 1995.

CAD Tools for the Development and Reuse of Models of Signal Processing Software and Hardware

Geoff Frank, Research Triangle Institute
Jim Armstrong, Virginia Polytechnic Institute

Besides developing software partitioning and VHDL test bench generator tools, RTI and VPI will be developing a signal processing algorithm library and a VHDL module library. The capabilities of these algorithms and modules will be analyzed, refined, and demonstrated. Furthermore, these algorithms and VHDL models will be interfaced to the tools in the RASSP design environment.

Populating VHDL Libraries for RASSP

Vijay Madisetti, Georgia Tech

This project will develop libraries of VHDL models for digital electronic macrocells, components, sub-systems and systems. The end product will be an extensive library that includes commercial-off-the-shelf (COTS) parts, digital signal processors (DSP) , a VHDL math library, and tools to aid in the generation, maintenance and standardization of VHDL libraries for RASSP. Our VHDL models for i860 and VME are being used extensively by the RASSP primes in their virtual prototyping demonstrations (which were also presented at the RASSP workshop in 1994), and models for the ADSP 21060 (SHARC,) PowerPC, TMS320C30, and other COTS and DSP chips are currently in development. All models are being collected into a respository for rapid dissemination, once they are validated and released. Technical point of contact: Dr. Vijay K. Madisetti, (404) 853-9830, email: vkm@ee.gatech.edu.

The Benchmark 1 Executable Requirement

Allan H. Anderson
MIT Lincoln Laboratory

The first RASSP benchmark, which was delivered to Martin Marietta and Lockheed Sanders by MIT Lincoln Laboratory in early August, included a tape cartridge with about 1/2 Mbyte of data and source code for an executable specification for a Synthetic Aperture Radar processor. The data is an accurate description of the signal transformations which the processor is to perform and it's system environment.

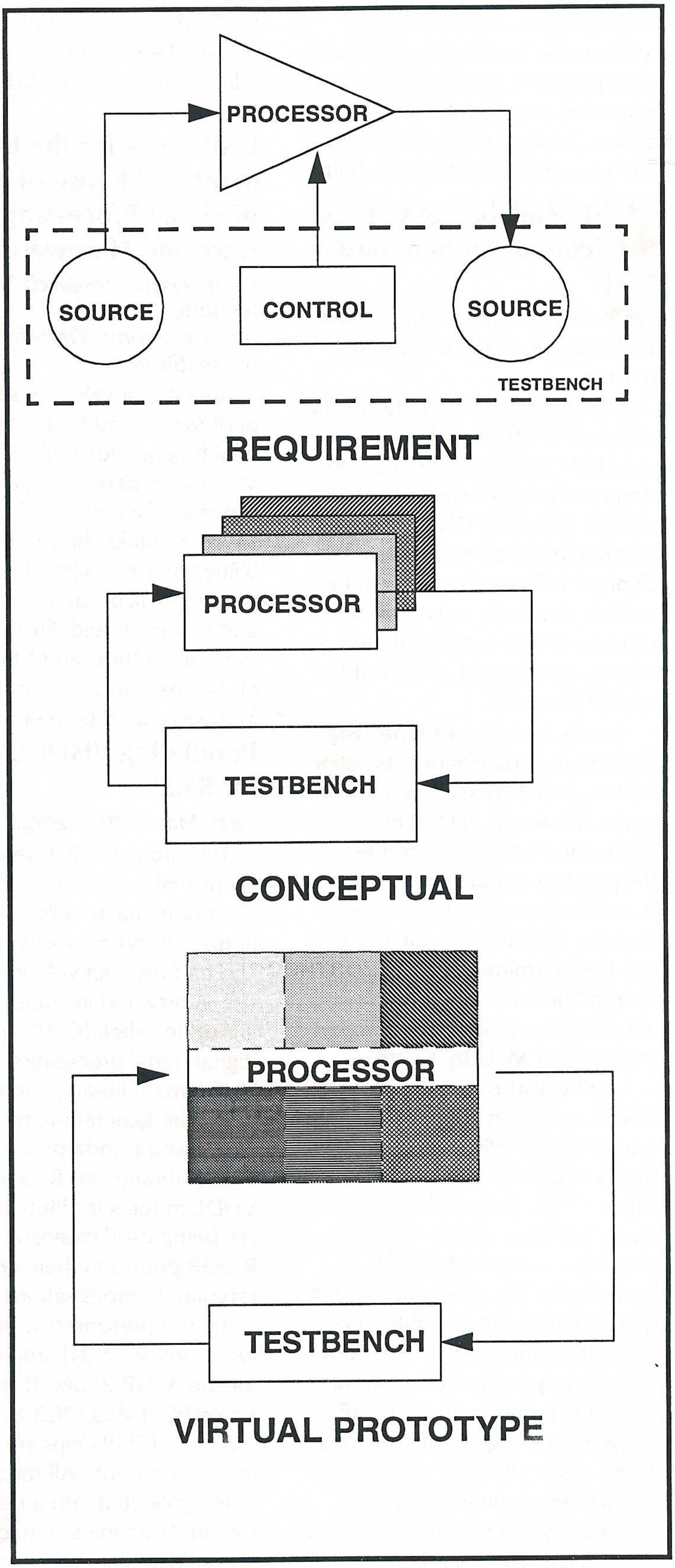
The use of executable specifications is essential to the RASSP Model Year concept which seeks to ensure that a signal processor will employ state-of-the-art technology when fielded and that it will be possible to upgrade the system throughout its lifetime. It is also a key to achieving improved design time and quality because it can provide a thread of evolving models from system definition to implementation.

What constitutes an executable specification and how to name the different varieties is a subject of active discussion, but common to all definitions is simulation of the processor in its environment. A design process can be thought of as a successive refinement and adding of detail to a processor model beginning with initial requirements and ending with a virtual prototype which models the hardware and software system in complete detail. An example of executable model evolution would include at least the three levels shown in the figure on page 16. They are:

1. Requirement - a simulation of the required signal transformations, with timing, in the environment in which the processor will operate. As shown in the figure on page 16, the environment includes a signal source, a controller and a data sink which may perform comparisons with a desired output. These comprise a testbench for the processor which will be used throughout the development to ensure conformance with the requirement.
2. Conceptual - simulations with different algorithms and performance models for exploring design trade-offs. Each conceptual model may contain modules at different levels of detail.
3. Virtual prototype - the final model with the processor partitioned into software and hardware modules. The virtual prototype's output will be identical to that of the final system and it is the definitive definition of the processor. Certain virtual prototype modules may serve as source for hardware synthesis.

Important advantages of interoperability and reuse accrue to use of one modeling language from requirement to virtual prototype but the needs at different levels are quite different. In the requirement the algorithm may not be specified in full detail. In conceptual models there is a high premium on fast execution time to improve designer productivity. And the virtual prototype must model hardware in detail and be capable of executing application code if the device is programmable. Languages and software environments such as Matlab, Processing Graph Methodology, C, Ptolemy, and VHDL are all candidates for executable specification languages. The optimum strategy for design with executable specifications is an important focus in the RASSP community.

The Benchmark 1 design exercise calls for creation of a virtual prototype for a real-time SAR processor for an existing radar, the MIT Lincoln Laboratory Advanced Detection Technology Sensor (ADTS). The ADTS radar is a fully polarimetric Ka-band radar installed in a Gulfstream G1 aircraft which



has flown about 400 data collection missions. The processor is specified so that it could be installed in the aircraft or on a UAV and create real-time images from ADTS data. At the peak rate of about three images per second a one-gflop/sec processor is required.

Since Lincoln Laboratory was working with an existing system this requirement has some of the characteristics of a system upgrade. The data input interface to the processor and data formats were completely determined and real data was available. There was some freedom for Lincoln in specifying the data output format and port and the control interface is a new design which is not completely specified to the two design teams.

The sponsors and Lincoln decided to create an all-VHDL Executable Requirement. The SAR strip map algorithm was implemented in VHDL through a straight forward translation of an existing C program. It primarily uses real and integer variables and VHDL signal variables very sparingly and executes the algorithm in zero simulated time. The VHDL created strip maps are essentially identical to those created with the C program. Data timing is modeled at the processor data input and output ports and the user can set processor latency between 0.1 and 3 seconds.

The VHDL testbench simulates the sensor system output by reformatting data from disk files and presenting it to the processor at the proper simulated time. It presents commands and setup data to the processor as a simulated host and writes output data from the processor to files and compares it with other disk file data. Latency is measured and compared with a user supplied reference. The processor and testbench model comprise 2430 lines of VHDL and use an existing math library.

The VHDL processor simulation is about 25 times slower than a C program for the math parts of the SAR algorithm, that is, an FIR filter, FFTs and vector multipliers. However, for the entire simulation with I/O, data reformatting and the small amount of timing simulation, the VHDL simulator is about 200 times slower than the C program which does no timing simulation. To simulate one frame of 512 pulses, for one polarization, in the Vantage Spreadsheet simulator requires about 2 1/2 hours on a Sun SPARC 10/51 workstation. This running time is acceptable for doing a requirement simulation but not when the testbench is used with processor models for conceptual design. Lincoln Laboratory is doing further work to identify any inefficiencies which may be contributing to run time.

The simulation was written to run on any IEEE STD 1076-1987 compliant simulator. Successful runs on Vantage Spreadsheet, Mentor QuickVHDL and Cadence Leapfrog simulators at Lincoln Laboratory and Lockheed Sanders, Martin Marietta and Wright-Patterson Air Force Laboratory, respectively, are proof of success. The simulation runs were done on different, but similar, uniprocessor workstations and no dramatic differences in run time was observed.

The delivered testbench will be used to exercise the virtual prototypes created by the developers. The second benchmark will probably be a prototype hardware implementation of the SAR processor and for that development Lincoln Laboratory is building a hardware testbench which will source and sink data in real time.

Vive La Difference

Vijay Madisetti
Georgia Institute of Technology

The erudite readers of The RASSP Digest may be curious as to the differences between the "current design process (circa 1993)" as opposed to a newer RASSP-like design process. Indeed, this editor has put much thought to this issue given the relevance, importance, and timeliness, of such a comparison. Some preliminary ideas that emerged from months of exposure to various facets of RASSP and also to current design practice are presented in the next few sections for your consideration, where this editor compares and contrasts two different (radically?) design processes. In the interest of space, high-level differences in design flow will be the focus of this discussion, as opposed to more involved architecture- and methodology-dependent facets that are well documented elsewhere. Thus, this editorial concerns itself with one (iterative) pass from concept to product. Higher-level loops that iterate over model-years will be the subject of future articles. We may recall that RASSP is targeted towards the design and prototyping (from concept to product) of large embedded DSP systems (we do not target ASICs as their technology is somewhat mature and sufficient resources are already addressing their rapid design and realization). Examples of systems of interest range from efficiently packaged single-board embedded DSP systems (as found in high-performance workstations using MCM-based chassis) to large multi-chassis STAP radar signal processor systems which typically have performance requirements ranging between 20-1000 BFLOPs of computational intensity at pixel rates of 10 Mhz, within the form constraints of size, weight, and power of 2-50 ft^3, 100-1400 lbs, and 1-10 KW, respectively. Boards represent subsystems, while multiboard configurations can represent complete systems, and involve hardware fabrication, assembly, and integration with application, control and diagnostic software. Figure 1 (listed below) represents a high level depiction of current design practice, and Figure 2 on page 19 represents a preliminary RASSP design flow. Both process flow diagrams start at the level of the representation of the application requirements. The algorithm to be implemented (e.g., a STAP radar signal processor system) is specified in an executable form (a VHDL/Ada, or a C/Matlab program) together with stimuli and test benches. In addition, the system has certain performance characteristics and constraints that must be met by the prototype (representative values being given in the preceding paragraph). After an appraisal of the application characteristics is completed, a partitioning of the application onto hardware (HW) or software (SW) is carried out manually by an experienced hardware system designer, and is to some degree ad hoc. Those portions of the systems that are to be cast as ASICs are selected, and common-off-the shelf (COTS) components such as processors and memories are chosen as targets for mapping SW components, and inital estimates as to the allocation of these parts are drawn and reviewed. The application is partitioned into subsystems (boards) so that each of these boards executes a portion (in SW or HW) of the algorithm, and their ensemble (the multi-board system) will hopefully prototype the required radar system with satisfactory performance. Since software (SW) cannot execute without target hardware, application and control software developed for each of these boards can only be tested and debugged after the hardware fabrication (assembly) and test of the board is completed (which can take 2-4 months per board, that too if complex ASIC design is not involved). After the hardware board is fabricated, application and control code is debugged in an iterative design cycle a of Figure 1. After successful design and test of the board-level HW/SW subsystem, the multi-board system is integrated manually, wherein the software and the hardware are merged and tested

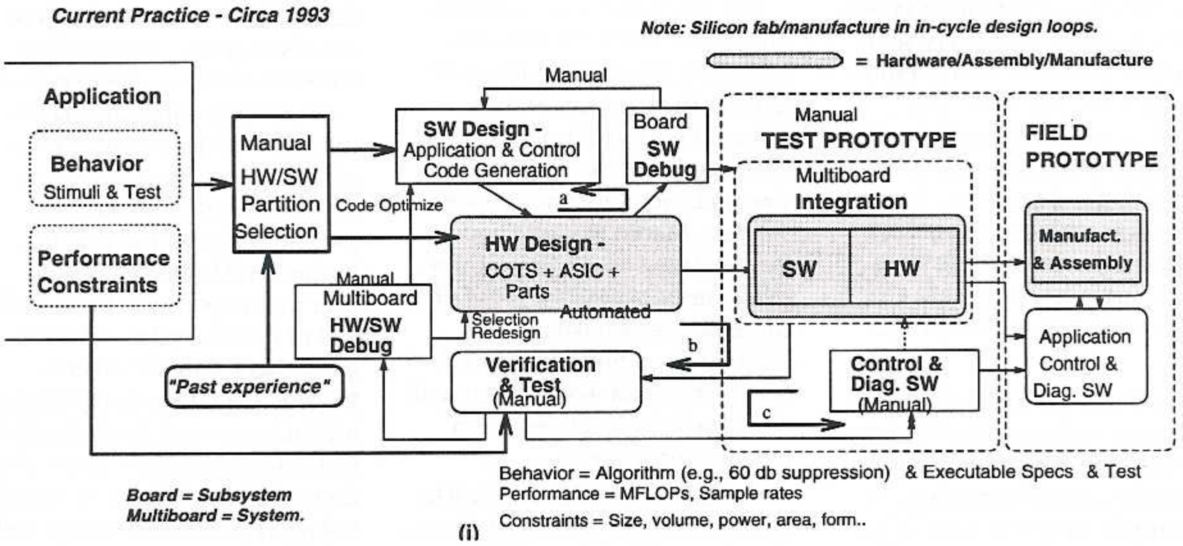
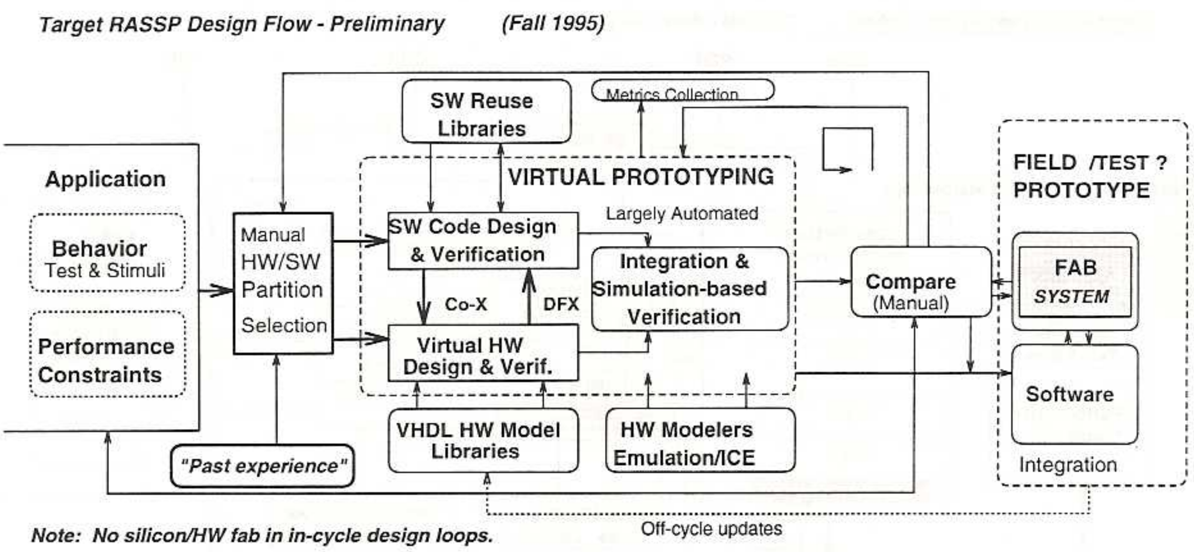


Figure 1
The author's vision of a "current practice" (circa 1993) design flow showing dependence between the software and hardware design and development cycles, with silicon fabrication and assembly/test (shaded regions) incorporated into three design loops (a, b, c)



Note: No silicon/HW fab in in-cycle design loops.

(ii)

Figure 2
The author's vision of a Preliminary RASSP Design Process (Fall 1995). Note the high degree of automation, removal of hardware fab from in-cycle design iterations, automated metrics collection, in addition to creation, maintenance and validation of VHDL libraries. Test prototyping (in silicon) is included as an option, and its selection depends on the accuracy of the modeling efforts.

via diagnostic software and input from the application (stimuli and test). This integration is done manually and involves silicon fabrication, manufacture and assembly/test, and is iteratively refined until an acceptable test prototype is synthesized. The three software design loops a, b, and c all include hardware fabrication, and the final field prototype is realized after satisfactory integration and test, often involving a total concept to prototyping delay of 3-4 years, at the cost of 20-30 man-years. In one representative radar signal processor system studied by this author, the final HW count was about 150 boards incorporating a total of about 25,000 LSI/COTS components including an ASIC front-end filter and a 64-processor TMS320C30-based processing engine. The final SW count in lines of source code (LOSC) was 5K LOSC for the DSP/radar signal processor application, 30K LOSC for control, 60K LOSC for diagnostics, and 25K LOSC for functional and performance verification, representing a 20:1 ratio of system-level design code size to DSP application code size. Designers focussing on hardware/software codesign are requested to include in their codesign software related to diagnostics and functional verification in addition to application (algorithmic) code generation. In the preliminary RASSP design flow of Figure 2, the primary difference (from Figure 1) is that the hardware fabrication and assembly at the subsystem and system-level is eliminated from the in-cycle design loop. The software is run on virtual hardware (in the form of VHDL models or hardware modelers and emulators) long before any HW fabrication and assembly is begun. This "virtual prototyping" environment significantly speeds up the design cycle through the use of models at multiple levels of design abstraction in the constituent VHDL libraries. The board-level and the multi-board integration is simulated and tested, additional control and diagnostic software is developed and debugged entirely in a user-friendly software environment. If the model libraries were accurate, the next stage could itself be that of the field prototype. However, at least one RASSP prime has planned to include the actual hardware test prototyping stage within the preliminary RASSP design process to validate and improve upon the process of virtual prototyping. The preliminary RASSP process, however, retains the manual HW/SW partitioning block as observed by the reader. The advanced RASSP-like design flow (Figure 3), scheduled for late 1996-1997, is envisioned by this editor as incorporating an additional stage defined as "conceptual prototyping" which involves

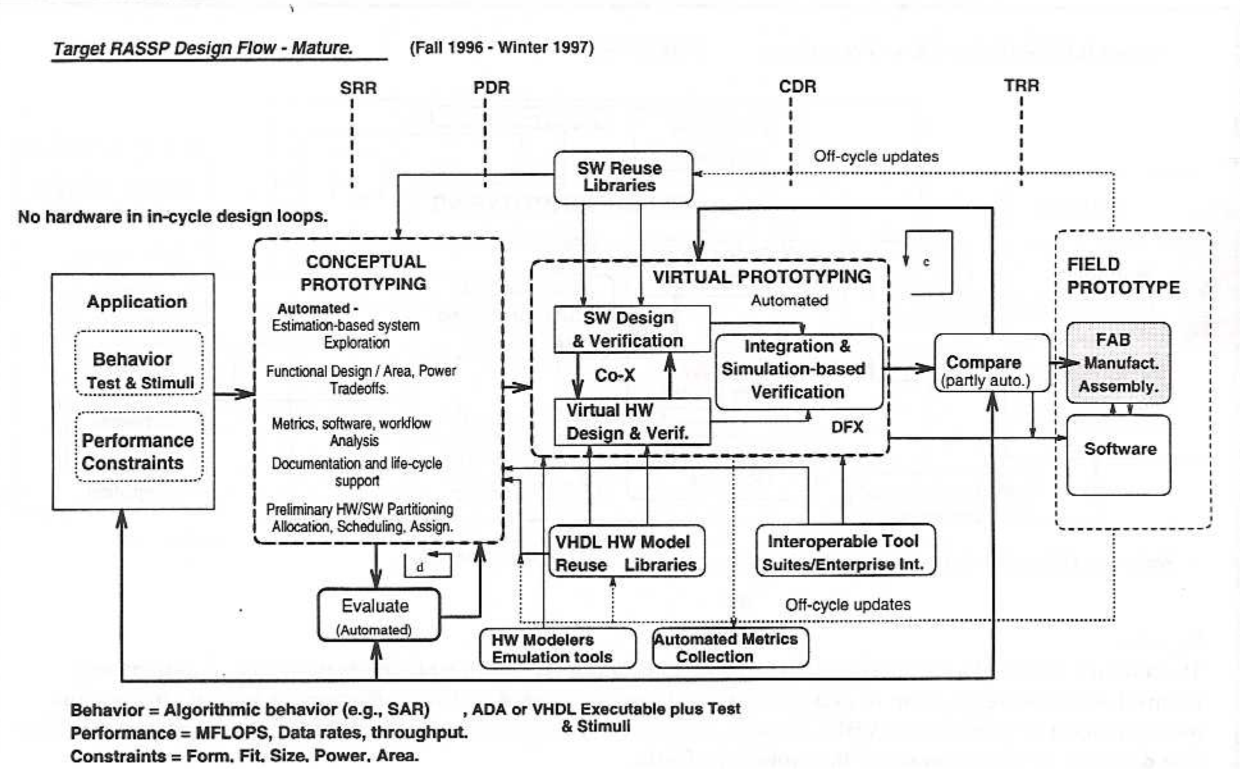

Target RASSP Design Flow - Mature.
(Fall 1996 - Winter 1997)
SRR
PDR
CDR
TRR
SW Reuse Libraries
Off-cycle updates
No hardware in in-cycle design loops.
Application
Behavior
Test & Stimuli
Performance
Constraints
CONCEPTUAL PROTOTYPING
Automated - Estimation-based system Exploration
Functional Design / Area, Power Tradeoffs.
Metrics, software, workflow Analysis
Documentation and life-cycle support
Preliminary HW/SW Partitioning Allocation, Scheduling, Assign.
VIRTUAL PROTOTYPING
Automated
SW Design & Verification
Co-X
Virtual HW Design & Verif.
Integration & Simulation-based Verification
DFX
c
d
Compare (partly auto.)
FIELD PROTOTYPE
FAB Manufact. Assembly.
Software
VHDL HW Model Reuse Libraries
Interoperable Tool Suites/Enterprise Int.
Off-cycle updates
Evaluate (Automated)
HW Modelers Emulation tools
Automated Metrics Collection
Behavior = Algorithmic behavior (e.g., SAR) , ADA or VHDL Executable plus Test & Stimuli
Performance = MFLOPS, Data rates, throughput.
Constraints = Form, Fit, Size, Power, Area.

Figure 3
The author's vision of a Mature RASSP design process (late 1996-1997) with no silicon in the in-cycle design loops, enterprise integration, interoperable tool suites, automated metrics collection, and an additional stage for rapid early algorithm, functional, architectural, and HW/SW partitioning in an automated manner called "conceptual prototyping". The design reviews (SRR, PDR, CDR, TDR) are also shown.

early design, and replaces the manual HW/SW partitioning block of the "current practice" of Figure 1. Conceptual prototyping utilizes automated tools that allow rapid estimation and evaluation of algorithmic, functional, architectural and enterprise-related trade-offs early in the design process. A few candidate conceptual prototypes are then culled from the dozen or so generated at this stage, and then passed on to the virtual prototyping stage. Here, extensive evaluation and detailed design is done in virtual hardware and software leading to successful and rapid integration, again through the use of HW/SW reuse libraries, interoperable tools, and enterprise integration. The entire process depends heavily on automation, and feedback currently being obtained from benchmark designs on candidate RASSP-like processes by the primes and other RASSP participants will be used to refine and improve upon both the rapidity, as well as the correctness of the first-time prototyping efforts of large DSP systems. The envisioned process presents a number of open problems related to both conceptual and virtual prototyping and verification that must be effectively addressed by various RASSP participants and the larger electronic systems design and application community, promising an exciting time for digital system designers trying to cut the prototyping times by a factor of four!. In summary, a RASSP-like process differs from current practice design process in the following ways: 1. No hardware fabrication, assembly, and test is present in in-cycle design loops. 2. Late binding of hardware allows the design product to be state-of-shelf at time of manufacture or use, 3. Extensive use of conceptual and virtual prototyping optimizes efficiency of the final product, and guarantees right-first time designs, 4. Design reuse supported by generation, maintenance, and upgrades of application-specific VHDL libraries for rapid design of signal processors, 5. Enterprise integration and interoperability between various point design tools facilitates design portability and standardization, 6. Extensive use of automation to facilitate -- a nested-loop and iterative design process, automated metrics collection and distributed collaboration facilities for large design project management speeds up the prototyping, a documentation and life-cycle maintenance process. comments.

Disclaimer: The views in this article are solely those of the author, and do not necessarily reflect the views of Advanced Research Projects Agency (ARPA), U.S. Department of Defense, Lockheed Sanders Inc., Martin Marietta Corporation, SCRA, or Georgia Tech.

Valuable feedback from Drs. Mark Richards (ARPA), Gary Shaw (MIT-LL), Dave Martinez (MIT-LL), Anthony Gadient (SCRA) and Jack Corley (SCRA) greatly improved the technical presentation and is acknowledged with gratitude. Please note that the next issue will address "VHDL and its use in rapid system development" and related RASSP efforts in this area.

Available Technical Publications

The RASSP Program: Overview and Accomplishments
M. A. Richards, ARPA
RASSP: Viewpoint from a Prime Developer
W. R. Hood, C. Myers, Lockheed Sanders, Inc.
Martin Marieta RASSP Program Overview
J. Saultz, Martin Marietta Laboratories
RASSP Benchmark Program Overview
G. A. Shaw, M.I.T. Lincoln Laboratory
RASSP Education and Facilitation
J. Corley et al, SCRA
RASSP Technology Base R&D Overview
J. Hines, D. Barker, Wright Laboratory
VHDL Performance Modeling
F. Rose, T. Steeves, T. Carpenter, Honeywell Technology Center
RASSP Methodology Overview
J. Pridmore, W. Schaming, Martin Marietta Laboratories
Processes and Experiences in VHDL Top-Down Design
R. Dreiling, Lockheed Sanders
VHDL Executable Requirements
A. H. Anderson, G. A. Shaw, C. T. Sung, M.I.T. Lincoln Laboratory
Test Bench Development for RASSP DSP Models
J. Armstrong, Virginia Tech; G. Franck, Research Triangle Institute
Design and Simulation of Heterogenous Systems using Ptolemy
B. Evans, A. Kamas, E. Lee, University of California at Berkeley
CAD Tool Interoperability Through Standards
D. Cottrell, J. Teets, CAD Framework Initiative
ADEPT: A Unified System Level Modeling Design Environment
S. Kumar et al, University of Virginia
Board and MCM Level Synthesis for Embedded Systems: The COMET Co-synthesis Environment
R. Vermuri, H. Carter, P. Alexander, University of Cincinnati
Applications of a Formal Model of VHDL
D. Benz, X. Fan, P. Wilsey, University of Cincinnati
Algorithms for Signal Processing
A. V. Oppenheim et al, Massachusetts Institute of Technology

Available Technical Publications (con't)

VLSI Discrete Wavelet Transform Architectures
K.K. Parhi and T. C. Denk, University of Minnesota
Adapting Algorithms to Architectures Through Transformations
G. Frank, B. Clark, W. Ransdell, Research Triangle Institute
Overview of the RACE Hardware and Software Architecture
B. Isenstein, Mercury Computer Systems Inc., B. Kuszmaul, Massachusetts Institute of Technology
Estimating the Requirements of Signal Processing Algorithms
B. Friedlander, University of California at Davis
Rapid Prototyping of Digital Systems with COTS/ASIC Components
S. Famorzadeh et al, Georgia Institute of Technology, R. Dreiling, M. Falco, Lockheed Sanders, Inc.
The Value of the Lockheed Sanders RASSP Approach
J. Trepanier, Lockheed Sanders, Inc.
Rapid Prototyping and the RASSP Design Environment (RDE)
J. Summers, Motorola
Image Signal Processor Demonstration
M. Vahey, Hughes Aerospace and Electronics Company
RASSP Technology Insertions
J. Pridmore, J. Evans, R. Graybill, Martin Marietta Laboratories
SAR Processing for RASSP Application
B. Zuerndorfer, G. A. Shaw, M.I.T. Lincoln Laboratory
Time Insensitive Binary to Binary Translation of Real Time Systems
B. Cogswell, Carnegie Mellon University
Z. Segall, University of Oregon
Predicting the Future with RASSP Benchmarks
J. C. Anderson, M.I.T. Lincoln Laboratory

For information regarding the availability of these publications, please send your request to info@rassp.scra.org

RASSP Digest-Rapid Prototyping of Application-Specific Signal Processors

The RASSP Digest is published quarterly and provides information for and about the RASSP Program and rapid systems development. For more information, contact Dr. Anthony Gadient or Dr. Vijay Madisetti, Editors, at: the addresses below.

Dr. Anthony J. Gadient
Phone : 803-760-4082
FAX: 803-760-3349
Email: gadient@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

Dr. Vijay K. Madisetti
Phone: 404-853-9830
FAX: 404-853-9171
Email: vkm@ee.gatech.edu
Georgia Tech
Sch. of Elec. & Computer Eng.
Atlanta, GA 30332-0250

Debbie Anderson
Managing Editor
Phone: 803-760-3792
Email: andersond@scra.org

RASSP Steering Committee

ARPA (ESTO)	
-Mark Richards	Program Manager
-Elliot Cohen	
ARMY (ARL/EPSD)	
-Clare Thornton	
-Randy Reitmeyer	Administrative COTR, Martin Marietta
-Arne Bard	Technical COTR, Martin Marietta
NAVY	
-Ingham Mack (ONR)	
-Gerry Borsuk (ONR)	
-Joe Killiany (NRL)	Administrative COTR, Lockheed/Sanders
-J. P. Letellier (NRL)	Technical COTR, Lockheed/Sanders
AIR FORCE	
-Bill Edwards	
-Stan Wagner	Technology Base and Facilitation/
-John Hines	Educator COTRs

Calendar of Events

VHDL International Users Forum *For More Information: VIUF* *(415) 329-0510*	*April 3-6, 1995*	*San Diego, CA*
32nd Design Automation Conference	*June 12-16, 1995*	*San Francisco, CA*
2nd Annual RASSP Conference *For More Information: Patricia Wolfhope* *(703) 351-8282* *Email: pwolfhop@sysplan.com*	*July 24-27, 1995*	*Arlington, VA*

SCRA
5300 International Blvd.
N. Charleston, SC 29418

RASSP Digest

RASSP - Rapid Prototyping of Application Specific Signal Processors

Vol. 2, No. 1, 1st. Qtr. 1995

Methodology

RASSP
Reinventing
Electronic
Design

Architecture Infrastructure

ARPA • Tri-Service

RASSP E&F
SCRA • GT • UCinc • EIT • UVA • Raytheon • MMG • ADL

In This Issue

Executive Outlook

Prime Development

Technology Base

Benchmark Program

Editor's Corner

RASSP and the Lockheed-Martin Merger

Mark Richards, RASSP Program Manager

Because the two RASSP prime contractors are Martin Marietta Laboratories and Lockheed Sanders, many people have asked about the effect of the proposed merger of the Lockheed and Martin Marietta Corporations on the RASSP program. After extensive discussions over the last few months, the Department of Defense, Federal Trade Commission, and the companies have agreed that the two RASSP primary development contracts will be continued essentially as is. Appropriate steps will be taken by the government and the companies to ensure that the two development efforts remain individually viable and competitive with one another.

Lockheed Sanders and Martin Marietta Laboratories were selected in the original RASSP competition because of their unique approaches to design methodology, digital signal processor architectures, and EDA infrastructure development. These attractive capabilities are not fundamentally changed by the proposed merger. Furthermore, the mechanisms established at the beginning by the RASSP program to ensure proliferation of RASSP design technology to the electronics design community at large remain in place and will still be effective after the merger. The continuation of both efforts provides the DoD the greatest possible breadth of technology development and impact on the EDA and defense supplier industries, and remains the best way to ensure the success of the RASSP program.

ARPA Manufacturing Technology Programs Ensure Military Access to Affordable Advanced Technology

by Mark Richards, RASSP Program Manager

(Note: portions of this column were adapted from a speech on technology partnerships given by Dr. Gary Denman, Director of ARPA in February of this year.)

As information technologies continue to become more capable, more compact, and more affordable, they will increasingly pervade forward deployed and mobile military systems. The Advanced Research Projects Agency's Electronic Systems Technology Office (ARPA/ESTO) has the charter to focus on electronic systems technology to produce the smaller, lighter, more mobile information systems needed by modern warfighters.

One of ARPA's technology investment areas which supports these goals is the electronic modules area. The Rapid prototyping of Application Specific Signal Processors (RASSP) program is an important component of the electronic modules program, but it is only a part of the story. Other ESTO programs in electronic module technology include Physical Electronic Packaging, Multi-Chip Integration (MCI), and Application Specific Electronic Modules (ASEM).

The Physical Electronic Packaging and MCI programs are developing multi-chip module (MCM) technology for digital systems operating at clock rates from 100 MHz to several GHZ, along with an order-of-magnitude reduction in manufacturing cost, development of a domestic supplier infrastructure, and acceleration of the acceptance and insertion of advanced multi-chip integration technologies. MCM technology offers the potential of 10-100X improvements in density, as much as 2-3X reduction in power, 10X improvement in reliability, and reduced cost.

The ASEM program strives to ensure the existence of an end-to-end capability to rapidly acquire electronic modules and subsystems. The program integrates and builds on the domain-specific building blocks such as physical packaging technology, packaging computer-aided design (CAD), flexible manufacturing processes and equipment, computer-integrated manufacturing (CIM), intelligent tests, design, interface, and test standards. A recent addition to this program was the establishment of a quick turn-around, semi-custom foundry for SEM-E format printed circuit boards populated with MCM technology.

RASSP represents the next step up in the "food chain" in this progression of ESTO programs aimed at improving the ability to design, package, and manufacture electronic modules. A new ESTO program in electronic systems packaging currently under consideration will continue the progression. In combination with programs of other ARPA offices in devices and circuits and in computing architectures and communications, these programs address the U.S. electronics industry's capability to develop complex, state-of-the-art electronic modules all the way from materials to system-level architecture.

It should be evident from these descriptions that manufacturing and affordability concerns pervade ARPA's electronic modules programs. In fact, they are a critical component in all of ARPA's planning. In the last several decades, the defense industrial base has become more and more isolated from the national or commercial industrial base, but today this defense-unique industrial base is shrinking. Furthermore, the most advanced technologies are no longer emerging exclusively through Department of Defense (DoD) investment, nor is the DoD any longer the dominant customer for most high technology. The United States must move toward a national technology and industrial base that will serve military as well as commercial needs. This strategy will allow DoD to exploit the rapid rate of innovation and market-driven efficiencies of commercial industry to meet defense needs, thereby achieving access to leading-edge technology, affordable products, and the ability to rebuild military capability should the world situation call for it. Developing technologies, components, and subsystems today that leverage commercial know-how will make the DoD stronger and more capable of meeting our national security needs of the future.

And yet manufacturing technology, which includes the electronics module program and RASSP in particular, is one of several ARPA programs that have recently come under fire as not relevant to national security. Others include SEMATECH, Electronics and Materials programs, Advanced Simulation, and Computing and Communications Systems. On the contrary, these are historical ARPA programs that are part of the nation's most successful military high-technology operation and are more vital to national security today than ever before. ARPA has a long history of delivering leading edge technologies that have provided the military the technological superiority it needs to prevail in crises, and many of these technologies have proven to have commercial application as well. ARPA's focus on leveraging commercial investments and knowledge as part of its technology research and development program will help meet critical defense needs by breaking down the barriers between the commercial and defense industries. Industry-led commercial development as a spin-off of critical military technology development has an important consequence for present and future military budgets -- the broader the application of the technology, the lower the unit cost to the military.

Defense R&D dollars are carefully invested to satisfy military needs -- to promote lower costs and higher quality at increased performance. DoD maintains a strategy to do what it can to ensure U.S. commercial industry remains at the cutting edge in those technologies that are also critical to our military capabilities. This necessarily requires DoD to support leading-edge research and development that accelerates the development of emerging commercial technologies that simultaneously meets defense needs.

A personal note: It has been my privilege to serve as the RASSP program manager for the last two years. In early May, I will be leaving ARPA for a new assignment. I am pleased to announce that Mr. Randolph (Randy) Harr of ARPA/ESTO will take over at that time as RASSP program manager and see the program to its completion. Randy has extensive experience in EDA, most recently at Stanford University and Synopsys. He will be a tremendous asset to RASSP, and I personally could not be more delighted to have someone of his caliber take over the program or more encouraged for its prospects under his guidance.

VHDL Modeling For Signal Processor Development

by Cory Myers and Ray Dreiling

Abstract

[1]This paper presents modeling approaches and experiences in the use of the VHSIC Hardware Description Language (VHDL) for the development of application–specific signal processors. Within our work on the ARPA/Tri–Service RASSP program we have developed and used VHDL modeling techniques for modeling the performance of signal processor systems and for the detailed design of signal processor systems. These approaches have been applied to modeling a large Infra–Red Search and Track system from a functional model, through performance modeling, through a full functional model, down to a detailed hardware implementation model.

1. Introduction

The ability of designers to rapidly develop and field application–specific signal processing is dependent on their ability to accurately model the systems that they wish to build. This modeling starts with modeling of the application to be developed, and it continues through architectural analyses and into detailed design.

Accurate modeling of the system being developed is key to good selection of architecture and to rapid development of hardware. Good models of hardware speed development by reducing errors in design and by allowing simultaneous hardware/software development.

The Rapid Prototyping of Application Specific Signal Processors (RASSP) program is an ARPA/Tri-Service initiative to create a new process for the development of military signal processors [1]. The objective of RASSP is to dramatically improve the process by which complex digital systems, particularly embedded signal processors, are specified, designed, documented, manufactured, and supported. The program is focused on the development of a process for the conversion of an initial set of requirements to an optimum signal processor architecture design and embodiment while simultaneously enhancing the ability to perform seamless upgrades as requirements change. RASSP is also addressing the obsolescence problems created by the inconsistences between the life cycles of major systems and their supporting technologies.

RASSP's objective is not just to support prototype development, but to support full–scale production and life cycle system support. This means that RASSP must support full military system design, including Ada, quantity production, and support. It must provide a mechanism for capturing the complete behavior of the system in a form that can be inexpensively maintained and upgraded for twenty years or more without compromising performance or safety. RASSP must support large teams working on large projects, fulfilling their needs while capturing the benefits of a “skunk works” project development style. Accurate modeling, at all levels of abstraction, from the functional to the detailed hardware level, is key to achieving these goals.

[1] The Lockheed RASSP team is under contract to the Naval Research Laboratory, 4555 Overlook Ave., SW, Washington, DC 20375–5326. The Sponsoring Agency is: Advanced Research Projects Agency, Electronic System Technology Office, 3701 North Fairfax Drive, Arlingbn, VA 22203–1714. The Lockheed RASSP team consists of Lockheed Sanders, Inc., Motorola, Hughes, and ISX.

2. Our Modeling Approach

Our RASSP design methodology is derived from the traditional top down design paradigm with the incorporation of the Virtual Prottype concept [2]. The basis of this concept is to develop a complete description, in standard languages like VHDL and Ada, prior to fabrication. The design is checked out completely as a model prior to commitment to hardware. In this way design errors are caught when they are easy to fix, and the system performance can be validated in simulation.

The choice of VHDL as the modeling language is important because VHDL provide:

Completeness: VHDL provides the mechanism for capturing the system behavior in a form that can be maintained and upgraded for twenty years or more; and

Portability: VHDL is an industry standard so models developed in VHDL can be ported to a wide range of simulation environments and can be maintained over the system's lifetime.

Our basic modeling approach is illustrated in Figure 1. The modeling begins with a functional description and proceeds through a series of refinements to produce detailed hardware and software. During this refinement process, as a sequence of models are developed to model system function, system performance, system detailed behavior, and detailed system design.

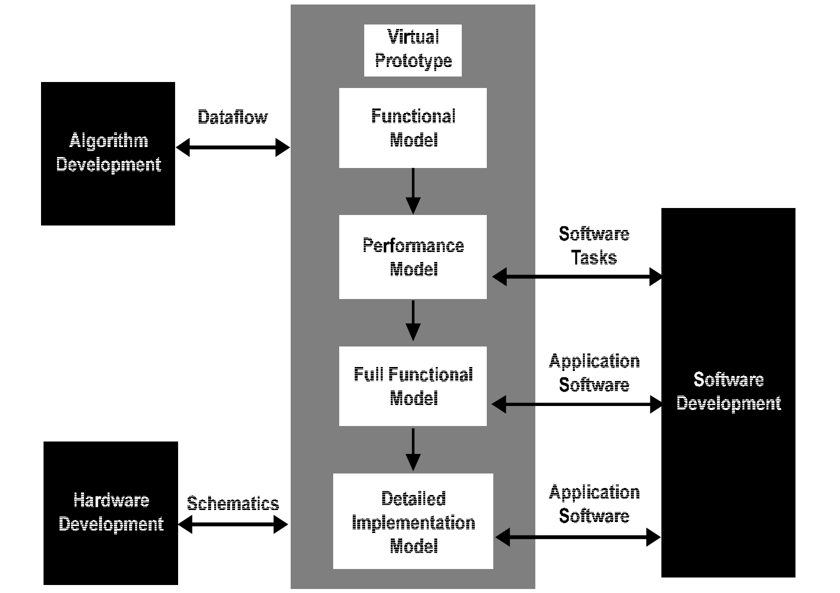


Figure 1. Our approach to model development starts with a functional model and refines it into a set of hardware and software descriptions.

2.1. Functional Modeling

The functional definition phase produces a data flow model that defines the systems behavior as a set of interconnected sub-functions prior to hardware/software partitioning. These sub-function models are either used directly or translated for reuse in lower level definition phases. We have used VHDL modeling in the functional definition phase, particularly in the context of the design of a SAR image processor for purposes of benchmarking the RASSP Process [3]. The use of VHDL modeling for a functional specification of a signal processing algorithm is unusual. It has the following advantages:

Consistent Testing Environment: Our design process is based on VHDL modeling so the functional definition is captured in the same form that the hardware development will be captured in. This allows for later side by side comparison between the functional representation and the detailed hardware design within the same environment.

Path to Synthesis: For those portions of the functional specification which will be implemented in custom hardware, rather than in a programmable processor, the VHDL description provides a better starting point for the hardware synthesis problem.

2.2. Performance Modeling

The Performance Modeling phase examines candidate architectures for trade-offs in both hardware/software partitioning and architectural elements. Performance models incorporate abstract application software descriptions. We have used a set of VHDL performance modeling libraries developed for the Air Force [4]. This library defines five basic types of architectural elements, as follows:

Pipelines: These architectural elements model devices which implement first–in/first–out behavior. Their behavior is primarily characterized by a delay.

Memories: These architectural elements model storage. Their behavior is characterized by their access time.

Bus Interface Units: These architectural elements model busses. Their behavior is characterized by a transmission rate.

I/O Devices: These architectural elements describe sources and sinks of data. They are characterized by their data rate.

Processors: These architectural elements describe programmable processors. They are characterized by their scheduler and by parameters of a set of resources.

An architecture is defined by connecting the architectural elements and mapping functional pieces to either dedicated hardware elements, which are modeled as pipelines, or to programmable processor elements. Functions mapped to programmable processors are modeled by specifying an abstract description of their resource and memory usage requirements. For example, an FFT implemented on a programmable processor may be characterized as requiring a certain number of floating point operations and a certain number of memory references. Characterization of processing elements and algorithms can be made data–dependent in a limited manner but the basic piece of data that flows between modeling elements, *a token*, consists of a marker that data has been produced, not the content of the data.

Architectural performance of mapping functions to architectural elements is determined by running the performance model and recording statistics about processor utilization, bus utilization, processing latency, memory usage, throughput, etc. Performance mod-

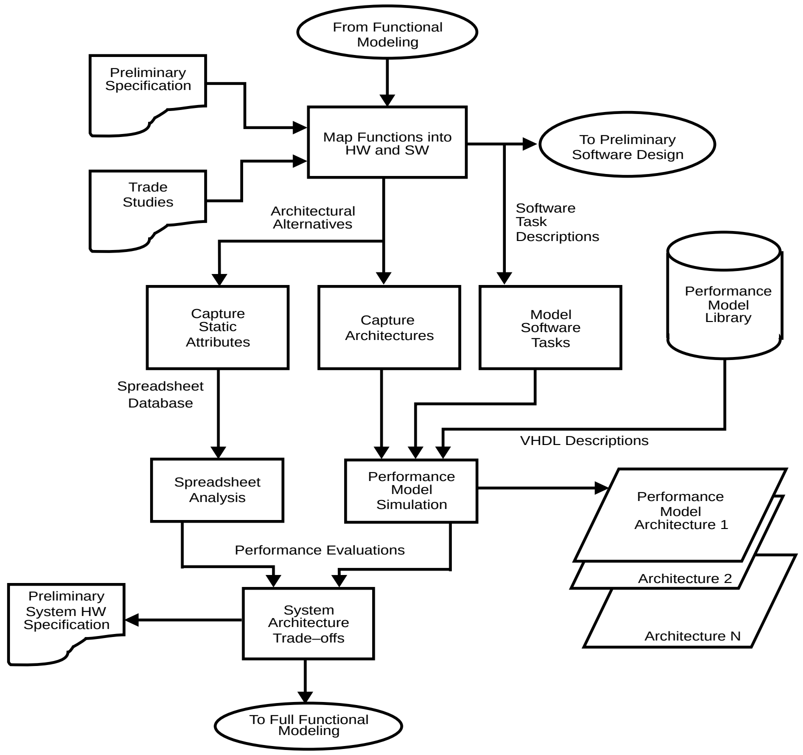

From Functional Modeling
Preliminary Specification
Trade Studies
Map Functions into HW and SW
To Preliminary Software Design
Architectural Alternatives
Software Task Descriptions
Performance Model Library
Capture Static Attributes
Capture Architectures
Model Software Tasks
Spreadsheet Database
VHDL Descriptions
Spreadsheet Analysis
Performance Model Simulation
Performance Model Architecture 1
Architecture 2
Architecture N
Performance Evaluations
Preliminary System HW Specification
System Architecture Trade–offs
To Full Functional Modeling

Figure 2. Development of architectural alternatives involves performance modeling of system.

eling is used in conjunction with static analysis, as illustrated in Figure 2.

2.3 Full Functional Modeling

A Full Functional Model provides a model that is structurally correct and exhibits the functional and performance characteristics of the entities being modeled. At this level dedicated hardware elements are modeled by their behavior, not in a way that implies their implementation. For example, a dedicated filtering chip would be modeled in way that was correct bit–wise but did not imply the implementation structure.

At this level of modeling, we have been using Instruction Set Architecture (ISA) and Instruction Set Simulator (ISS) models. Our ISA modeling approach is to develop VHDL behavioral models of a processor that can execute software and provide complete access to the internal registers of the processor [5]. Our ISS modeling approach is to integrate a commercial processor simulator into a VHDL environment. In either case, the processor model is combined with a Bus Interface Model (BIM), which models the detailed interaction of the processor at its connections, to make the full functional model. We have used this approach both to model individual chips, i.e., the i860, and to model single board computers.

Both the ISA and the ISS approaches allow the application software that is to run on the target hardware to be run in the simulation environment prior to physical hardware delivery. It is at this stage in the modeling process that detailed errors about the meaning of interfaces can be identified and corrected. Additionally, this type of model gives the software much more accessibility to the state of hardware than is often the case when the software is run on the physical hardware.

2.3. Detailed Implementation Modeling

The Detailed Implementation Modeling provides models which are sufficient to determine the implementation of components. For example, at this level the exact structure of multipliers, adders, and registers would be obvious for a digital filter. This detailed model is derived from the abstract behavioral model by a combination of synthesis tools and manual design.

It is our approach to the modeling problem to only develop detailed implementation models for those components which we are developing. For components that are purchased, we use the abstract full functional model. Thus, a complete system simulation is a mixed level model.

3. Results

Our experiences with the VHDL modeling approach includes development of an Infra–Red Search and Track (IRST) processor and a Synthetic Aperture Radar (SAR) processor [6,3]. Both processors have been developed from functional specification to hardware designs using the modeling approach. We will describe our work on the IRST processor in more detail here. Our work on the SAR processing problem will be completed by mid–1995.

3.1. The IRST Processor Problem

The IRST processing problem is illustrated in Figure 3. It takes input data from an infrared sensor at between 15 to 135 million samples per second depending on the sensor size. Input data is in fixed point format. The output rate is small, being just detected target message reports and graphic symbology. The image size is 2200 by 1800 pixels and processing takes place most in overlapping subimages of about 200 by 200 pixels.

The processing required for the IRST problem is a function of the sensor coverage, the scene revisit interval, the type of filtering algorithm, and the number of target movement hypotheses. Requirements can range from half a million to several hundred billion operations per second (BOPS). For our problem we require 5 BOPS.

3.2. Modeling for the IRST Problem

For the IRST problem we have developed a working system that is to be flown aboard a test aircraft. In terms of our modeling process we performed the following steps:

Functional Modeling: We developed a VHDL model of the IRST processing algorithm by translating an existing C code implementation.

Performance Modeling: We used the performance modeling approach to analyze three candidate architectures for the IRST processing problem. We examined the Mercury Raceway architecture, the Intel Paragon MP, and the ISI Embedded Variant and were able to determine that the most appropriate architecture for this problem was the Mercury Raceway (due to communications issues). The

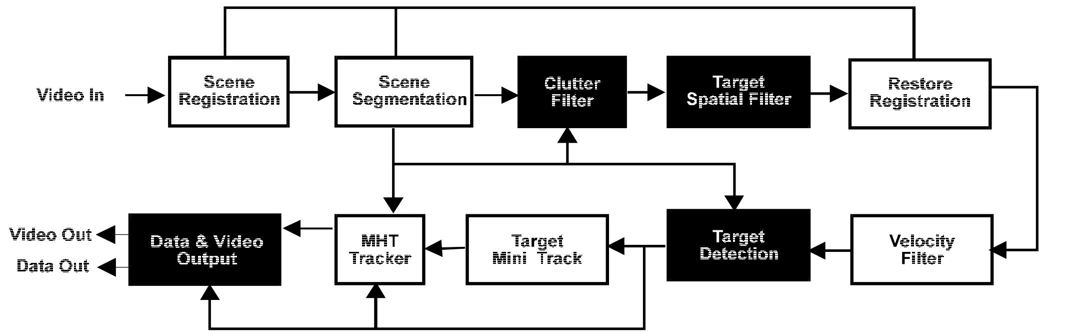


Figure 3. The IRST processing problem requires target detection and tracking. The shaded steps are required processing while the others are required only in heavy clutter environments.

resulting architecture consisted of a Mercury Raceway with multiple MVC9 boards (16 i860 processors per board), dual video input cards, a video output card, and a system controller, as illustrated in Figure 4.

Full Functional Modeling: We developed a full functional model of the MCV9 board using an ISA model of the i860. We also developed behavioral models of the video input and output boards. Driver software was run on the i860 models and was used to control the video input and output models.

Detailed Implementation Modeling: We developed detailed implementation models for the custom components (FGPA programming) on the video input and output boards.

The total VHDL model consists of over 60,000 lines of code (LOC), including 4000 LOC for the i860 ISA model, 4000 LOC for Mercury's custom processor connection ASIC, 4500 LOC for Mercury's custom interconnect ASIC, and 5300 LOC for the VME logic. These models were developed by a team of eight people, geographically distributed among Lockheed Sanders, Hughes, and Motorola, over a period of nine months. The model is capable of processing 1500 simulated instructions per second for the ISA model using the Vantage VHDL simulator on a Sparc 10/40 workstation.

3.3. Lessons Learned in the IRST Problem

During this modeling process we have observed the following strengths in the modeling process:

Common Language: The use of VHDL performance modeling allowed the system engineers, the hardware engineers, and the software engineers all to interact on a common, executable, model of the processing problem. This eliminated many communications issues within the team.

Elimination of Hardware Errors: The development of a complete system model with the proper structure allowed the development team to catch and eliminate several hardware interface errors that would normally have been found after physical integration.

Early Software Debugging: The ability to run driver software on the behavioral model allowed system software debugging to start early. This process caught coding errors which otherwise would not have been caught until after physical integration. Additionally, the simulation environment has the potential to provide more access to what was happening in the hardware than is often found in the physical hardware.

Hardware/Software Codevelopment: The modeling approach allowed simultaneous development of hardware and software. Additionally, within the simulation environment the hardware and software engineers were able to easily negotiate the details of register formats and coding.

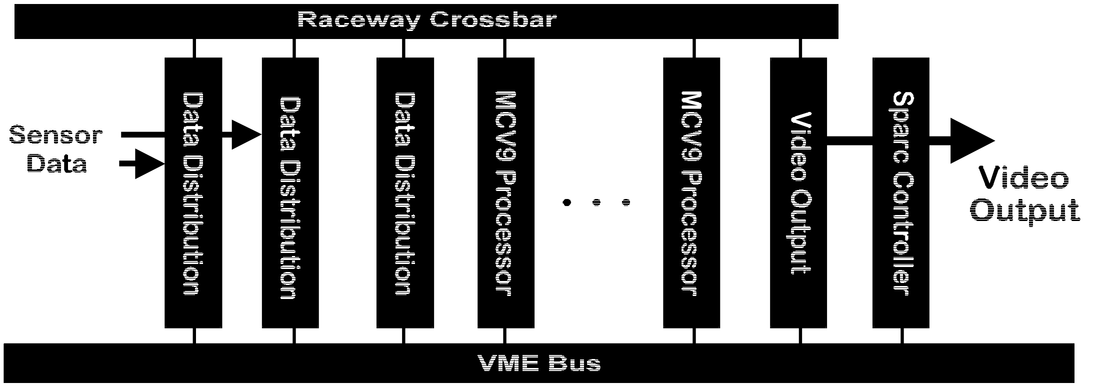


Figure 4. The IRST processor consists of video input, processing, video output, and a controller.

We have also observed some weaknesses in the modeling process, including:

Lack of Existing Models: Development of the performance models, the ISA model, and the behavioral models were time–consuming efforts. We often had to develop our own full functional models for COTS parts. Fortunately, much of what was developed here can be reused on other problems.

Lack of Software Debugging Tools: Our use of the ISA model for a processor did not allow the use of standard debugging tools. We are working to solve this problem by using an ISS model in our current SAR work.

Maturity of the Performance Modeling Library: The chosen library of performance models was not the most mature. The RASSP program is addressing this issue by funding commercialization of the performance modeling technology.

Large Simulation Resources Required: The VHDL simulations required a Sparc 10 workstation with 256 Mbyte of memory and 500 Mbyte of swap space. Simulation runs typically produced simulation files of 200 Mbytes. We are working to reduce these requirements by investigating alternative VHDL simulation technologies.

4. Summary

In summary, we have presented a top–down approach to developing a complete VHDL model of a signal processing system. This approach has been used successfully for modeling the development of an IRST processor for our RASSP demonstration project.

REFERENCES

[1] J. Corley, V. Madisetti, and M. Richards, “Introduction to ARPA's Rapid Prototyping of Application Specific Signal Processors (RASSP) Program,” in the Proceedings of ICASSP 1995.

[2] R. Dreiling, “Processes and Experiences in VHDL Top Down Design,” in the Proceedings of the First Annual RASSP Conference, August 1994.

[3] B Zuerndorfer and G. A. Shaw, “SAR Processing for RASSP Application,” in the Proceedings of the First Annual RASSP Conference, August 1994.

[4] Honeywell SRC, “Graphics Processor Definition VHDL Processor Model,” AF Contract F33615–90–C–3800.

[5] S. Famorzadeh, T. Egolf, V. K. Madisetti, P. Kalutkiewicz, M. Falco, and R. Dreiling, “Rapid Prototyping of Digital Systems with COTS/ASIC Components,” in the Proceedings of the First Annual RASSP Conference, August 1994.

[6] M. Vahey, “Image Signal Processor Demonstration,” in the Proceedings of the First Annual RASSP Conference, August 1994.

For more information on VHDL Modeling For Signal Processor Developmnet, contact Cory Myers at cory@sanders.com

Architectures for Rapid Prototyping of Embedded Signal Processors

by G. Caracciolo and J. Pridmore

Abstract

The Rapid Prototyping of Application-Specific Signal Processors (RASSP) program is striving to change the way embedded signal processor design is performed, providing >4X improvements in time-to-market, cost, and design quality. These improvements will be achieved using a methodology that stresses hardware and software reuse in conjunction with Model Year Architectures that facilitate reusability and upgradeability through open interface standards. This paper will describe a Model Year Architecture approach for the development of cost-effective signal processors that can be applied to a wide range of military and commercial applications.

1. Introduction

The drivers for RASSP signal processor architecture definition result from the requirements imposed on signal processors to meet changing mission-critical processing needs and military requirements for long-term life cycle support. Additionally, RASSP must address the full spectrum of signal processing applications, from low-cost commercial applications, such as cellular communications and HDTV (1-10 processors), to very large military sensor systems, such as shipboard radar systems (100 - 1000 processors). This range of requirements imposes a formidable challenge in defining an architectural approach that addresses low-cost technology insertion, upgradeability, and extensibility.

The Model Year Architecture (MYA) is being developed to address these issues, promoting design upgrades and reuse via standardized, open interfaces, while leveraging state-of-the-art commercial technology developments. Designs are performed using a concurrent engineering process that facilitates continuous product improvements via iterative virtual prototypes, which can be easily retargeted to support a range of applications[1]. Model Year architectures must support scalability, heterogeneity, open interfaces, modular software, life cycle support, testability, and system retrofit.

RASSP Model Year architectures must be supported by library models to facilitate trade-offs and optimizations for specific applications. The hardware and software elements within the library are encapsulated by functional wrappers, which add a level of abstraction to hide implementation details and facilitate efficient technology insertion. Thus, the notion of Model Year upgrades is embodied in reuse libraries and the methodology for their utilization.

2. Model Year Architecture Framework

The RASSP program supports the design of architectures through a framework that provides a structured approach to ensure that designs incorporate all the required Model Year features described above[2]. The basic elements that comprise the MYA are the Functional Architecture, Encapsulated Library Components, and Design Guidelines and Constraints, as shown in Figure 1. Synergism between the MYA framework and the RASSP methodology is required, as all areas of the methodology, including architecture development, hardware/software codesign, reuse library management, hardware synthesis, target software generation, and design for tests are impacted by the MYA framework.

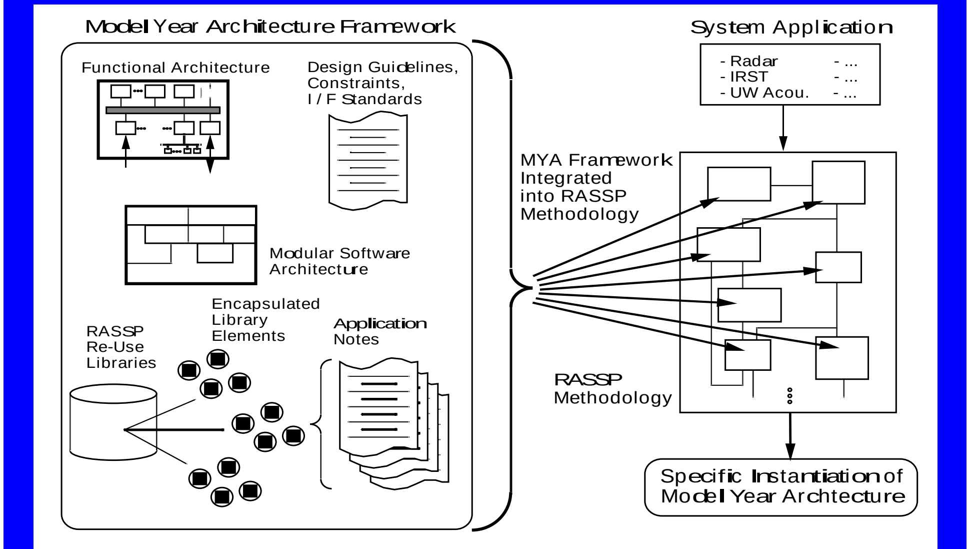

Model Year Architecture Framework
Functional Architecture
Design Guidelines, Constraints, I / F Standards
Modular Software Architecture
RASSP Re-Use Libraries
Encapsulated Library Elements
Application Notes
System Application
- Radar - ...
- IRST - ...
- UW Acou. - ...
MYA Framework Integrated into RASSP Methodology
RASSP Methodology
Specific Instantiation of Model Year Archtecture

Figure 1. Model Year Architecture Framework

The Functional Architecture defines the necessary components and the manner in which their interfaces must be defined to ensure that the design is upgradable and facilitates technology insertion. As such, the Functional Architecture is a starting point for developing solutions for an application-specific set of problems, not a detailed instantiation of an architecture. Specifically, the Functional Architecture specifies a high-level starting point for performing application-specific architecture selection; a standard approach for selecting and implementing standard, open interfaces; and guidelines for efficient verification and test. The Functional Architecture DOES NOT specify the topology or configuration of the signal processor architecture, specific processor types, or system-level interface standards (external to the signal processor).

The Functional Architecture concept is based on the use of abstract architectural objects and standard functional interfaces at key points within a layered architecture. An important aspect of the Functional Architecture is that application-specific realizations of a signal processor are embodied in the proper definition and use of Encapsulated Library Elements. Encapsulation refers to additional structure added to otherwise raw library elements to support the Functional Architecture and ensure library element interoperability and technology independence to the maximum extent possible. Incorporated within the reuse libraries are application notes that the designer can use to properly apply and aggregate the individual hardware and software components into a final processor product.

The MYA Framework also provides a set of Design Guidelines and Constraints for general architectural development, such as how to properly use the functional architecture framework, general use of encapsulated libraries, and most importantly, procedures and templates to encapsulate new library components. These design guidelines and constraints are incorporated into the RASSP design methodology.

The Model Year Software Architecture, shown in Figure 2, simplifies developing high-performance, real-time DSP applications allowing the developers to easily describe, implement, and control signal processing applications for multiprocessor implementations. The architecture supports the Model Year concept by providing a common Application Programming Interface (API) to the underlying real-time operating system services. This allows a new hardware platform with a new microkernel to change for each model year while maintaining the API. Support for the API is through the RASSP Run-Time System (RRTS), which provides the services required for the control and execution of multiple graphs on a multi-processor system. The RRTS and its support for the API forms the essential component of software encapsulation for a processor object.

The application layer is divided into two parts, similar to the Processing Graph Method (PGM) developed by the Naval Research Lab [3]. The first part of an application is the Command Program, which provides response to external control inputs, starting and stopping data flow graphs, managing I/O devices, monitoring flow graph execution and performance, starting other command programs, and setting flow graph parameters. The Control Interface provides services that implement these operations.

The second part of the application layer is the data flow graphs (DFGs), implemented using a data flow language. Services provided by the DFG interface are largely invisible to the developer and include managing graph queues, interprocessor

Figure 2. Model Year Software Architecture

communication, and scheduling. The RASSP program will support static and dynamic scheduling paradigms. The constructed flow graph will be converted into a HOL such as C or Ada via autocode generation and will contain calls to a standard set of domain primitives. A full suite of tools is being developed on RASSP to support this software architecture. All RASSP tools will be made commercially available.

3. Applying the Model Year Architecture Framework

3.1 Hardware Architecture

Verification of a MYA signal processor is iteratively performed throughout the codesign process, requiring the reuse libraries to support models at various levels of hierarchy. Three levels of VHDL modeling hierarchy are currently being developed and used in a series of benchmarking experiments to define reuse library elements for RASSP:

Performance/Uninterpreted/Architectural models provide timing-only behavior for processor nodes, buses/ interconnects, etc. to support high-level architectural trade-offs (number and types of processors, type and topology of network).

Abstract Behavioral Models provide full functional behavior at the data output level with (potentially) an abstract level of timing. This level includes both algorithm-level and Instruction Set Architecture (ISA)-level models.

Full-Functional and Bus-Functional models provide full functionality at the signal level and timing fidelity at the clock level. This includes Register Transfer Level and logic models.

Through these models, the Functional Architecture constructs are supported. For example, Figure 3 illustrates an application of a functional interface at the hardware level for a construct called a Reconfigurable Network Interface (RNI). The RNI is divided into three logical elements: 1) local interface, 2) external interface, and 3) bridge element. The local and external interfaces implement the specific protocols to the elements being interconnected, in this example a HIgh speed Parallel Port Interface (HIPPI) and VME interface. The bridge element, which typically consists of a buffer memory and a controller implemented via custom logic (e.g. FPGA, ASIC) or a programmable processor, performs the actual bridging function. The buffer memory facilitates asynchronous coupling and flow control between the two networks, while the controller coordinates data transfers. The three logical elements of the RNI are implemented as encapsulated library elements that serve to isolate changes resulting from upgrades. For example, the VME interface could be replaced by another encapsulated interface, such as the Scalable Coherent Interconnect (SCI), with little or no impact on the HIPPI hardware and software.

To refine details of various architectural constructs and to determine their performance impact, a number of experiments are ongoing, primarily through VHDL modeling and simulations. For example, i860 ISA-level models [4] and the FPASP5 vector processor model developed by Rome Laboratory [5] are being used in conjunction with models for Peripheral Component Interconnect (PCI) and HIPPI interfaces. These models are being used as vehicles for refining the functional architecture concept by encapsulating the models and demonstrating a plug-and-play capability among the i860, FPASP5, and the different interface elements. Note that the functional interface at the software level must also be maintained, which will also be verified by executing interface software on the simulation models.

3.2 Software Architecture

Software development cannot be discussed without its relationship to the architecture of the signal processor; in fact, it is an important part of the application-specific architecture design process. The representation of architectural elements as objects includes not only hardware representations in the form of VHDL models, but also behavior defined by the software libraries associated with that hardware. The software portion of

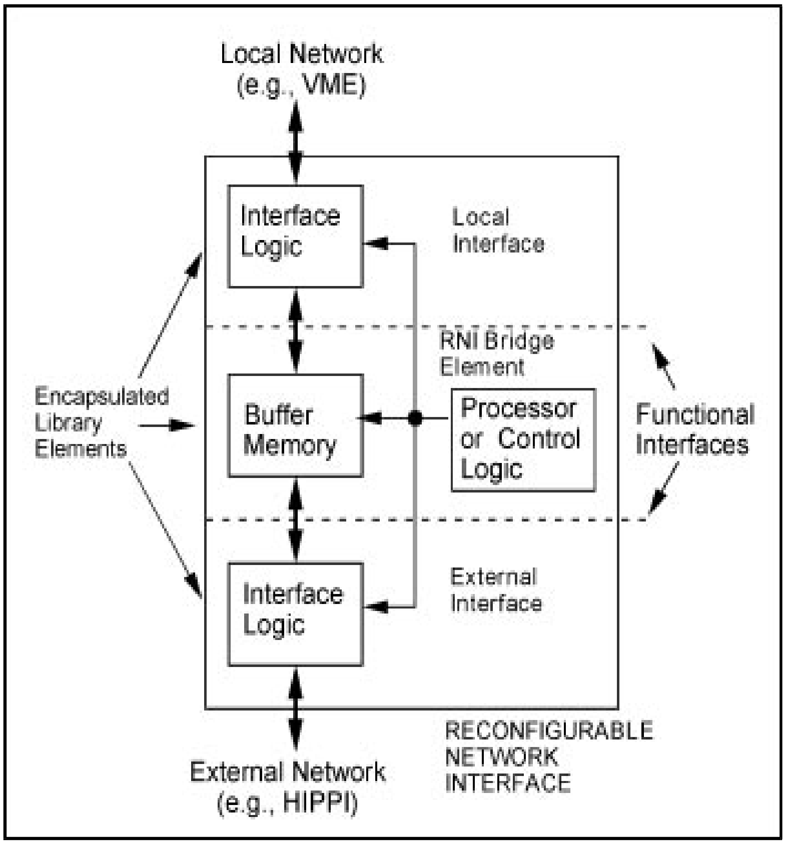


Figure 3. Functional Interface Example Applied to a Reconfigurable Network Interface

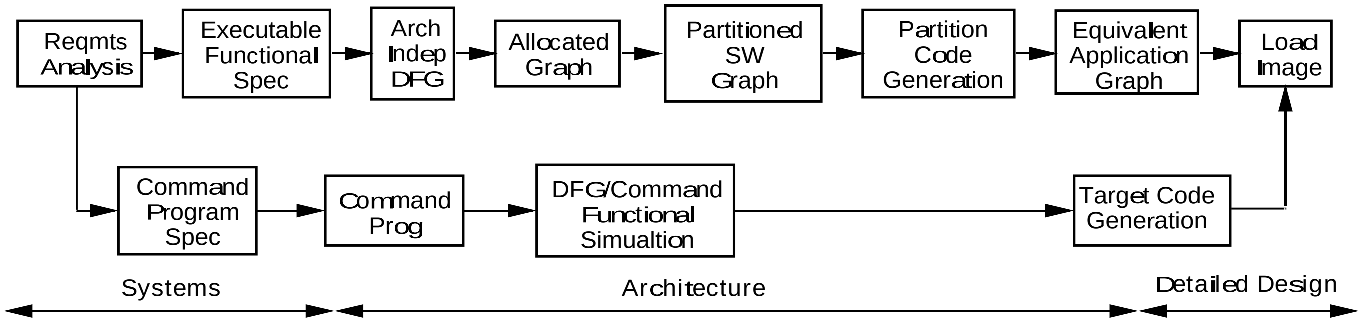


Figure 4. RASSP Graph-Based Software Development Scenario

architectural objects is handled by the process shown in Figure 4. This process depicts the progression of software generation from the requirements to load image, with emphasis on the graph objects involved and the general RASSP process in which they occur. It also shows the parallel development and co-simulation of the command program.

Architecture definition involves the creation and refinement of the data flow graphs that drive both the architecture design and the software generation for the signal processor. The data flow graph(s) of the signal processing are developed, and the nodes are allocated to either hardware or software. Automated generation of the software partitions is performed to provide executable threads that are to be run on the DSPs. These autocoded partitions are combined into an application graph which is functionally equivalent to the original. The graphs are co-simulated with the command program to ensure proper interaction.

The final step in the software development, which is the production of the load image, occurs during detailed design. The software load image generation is an automatic build process that is driven by the autocode generation results. The inputs to the process include the architectural description, the detailed DFGs describing the processing, the partitioning and mapping information, the autocode generation results, and the command program. The process is controlled by a software build management function which extracts the necessary information from the library and manages the construction of all the downloadable code as directed by the partitioning and mapping data.

This process is verified through virtual prototyping prior to committing to an actual hardware build and is carried out at several levels of hierarchy including performance level simulations, ISA level simulations of key hardware and software elements, and low-level simulation of hardware interfaces.

4. Conclusions

The RASSP program is applying a Model Year Architecture concept to the rapid prototyping of embedded signal processors. This concept facilitates reusability and regular, low-cost technology upgrades. This is accomplished through the definition of a framework for developing open architecture signal processors, which can be applied to a wide range of military and commercial applications. The framework relies heavily on Object-Oriented concepts to properly encapsulate the architectural reuse library components that are modular and scalable. Ongoing work is refining the concepts of the Model Year Architecture framework, including the definition of architectural object classes, interfaces, and attributes for the various elements. Additionally, benchmarks are being developed to quantify hardware and software overhead through virtual prototype examples to refine the encapsulation concept. The MYA will support an automated reuse-based code generation process for heterogeneous multiprocessors.

For more information on Architectures for Rapid Prototyping of Embedded Signal Processors, contact Jeff Pridmore at
jpridmore@alt.ge.com

ACKNOWLEDGMENT
The work reported here was supported by the ARPA/Tri-Service RASSP program, contract DAAL01-93-C-3880.

References

[1] Mark Richards, The Rapid Prototyping of Application Specific Signal Processors (RASSP) Program: Overview and Accomplishments, Proceedings of the First Annual RASSP Conference, August 1994.

[2] Gerald Caracciolo, RASSP Model Year Architecture Working Document Version 1.0, October 28, 1994.

[3] Naval Research Laboratory, Processing Graph Method Specification, Version 1.0 11 Dec. 1987.

[4] V.J. Madisetti, T. Egolf, S. Famorzadeh, L-R Dung, Virtual Prototyping of Embedded DSP Systems, to appear in Proceedings of IEEE ICASSP 95.

[5] Richard Linderman, Ralph Kohler, Designing a Wafer-Scale Vector Processor Using VHDL, GOMAC 1991 Digest of Papers. 1991 pp 65-68.

Honeywell Develops VHDL Performance Model Library

by Fred Rose, Honeywell RASSP Technical Director

1. Introduction

The design and development of high performance computing systems is becoming increasingly complex. A primary ingredient of a sound design methodology is a detailed performance model of the system. A performance model expressed in VHDL serves as a simulatable specification, aids the identification of bottlenecks, and supports performance validation. It also allows trade-off studies for what-if analysis and documentation of design decisions. Honeywell Technology Center (HTC) has developed a library and a corresponding methodology for developing VHDL performance models. These models can be used for capturing and documenting architectural level designs, and for doing performance analysis studies of the architecture. HTC is significantly enhancing this library for the Martin Marietta RASSP team.

Performance modeling is applied during the early stages of system development. It provides another tool to the system designer but is not intended to be stand-alone nor discarded at the end of the system development stage. Performance modeling can aid evaluation of design alternatives, capture design decisions and assumptions, examine system behavior at boundary conditions, and help determine bottlenecks and overdesign. The system designer can also utilize performance modeling for examining system sizing, topology, partitioning and capability issues. An important benefit of performance modeling is that it provides early interaction of system, hardware, and software designers.

HTC has developed a generic, parameterizable library of VHDL performance models. This library consists of input/output devices, memories, bus communication elements, and a processor model. The processor model is the key element to the performance modeling methodology as it facilitates hardware/software codesign and coanalysis. The processor model has the capability of modeling software tasks and scheduling.

This VHDL performance environment allows the systems architect to capture the system under study in a consistent, verifiable form. The VHDL simulation produces metrics which can be used by any commercial analysis package, spreadsheet, or other appropriate format to aid the design decision process. The results can be directly compared with the system specification to verify that the architecture meets the performance requirements. Once the architecture is verified (the latency, utilization, and throughput meet requirements, the system is self consistent, and size, weight, and power limits are met), the system is ready to proceed to detailed design. The performance model can also produce the architecture's characterization for its use in a higher level model, where this design would just be another building block.

The primary focus of HTC's RASSP tasks for Martin Marietta are to help facilitate interoperability at the system design level by developing a VHDL performance model interoperability guideline, and to provide a robust library to achieve the system VHDL performance modeling.

2. Performance Model Interoperability

For the RASSP program to truly realize the full benefits of performance modeling, a common modeling approach is required. VHDL is the Lingua Franca, that is the language for doing business, of the EDA world. As the EDA industry increasingly moves to a “plug and play" business model, a common language becomes essential. However in addition to a common language, common usage or style is required. This has been recognized from the beginning in the VHDL community and a variety of style related standardization activities have occurred. However, to date, these efforts have focused on lower levels of design abstraction. The performance model interoperability guideline will raise that level of abstraction.

It is critical that the performance models interoperate with the tools used to specify and capture the RASSP requirements and system design information. The interoperability guideline defines how Honeywell's performance model library will function with other elements of the RASSP environment. Honeywell will work with the primary architecture-level tool vendors, including JRS, Vista, and Omniview, on the Martin Marietta team, and additionally Lockheed-Sanders and the University of Virginia to establish interoperability with their tools and VHDL performance models. This interoperability guideline is meant to be a consensus opinion on VHDL performance modeling interoperability. Because Honeywell has a role on the Martin team to develop and support VHDL performance models, this document is initially heavily weighted towards that role. Hopefully as this technology becomes more widespread throughout the RASSP community, the interoperability guideline will become more generic.

The interoperability guideline consists of the following sections:

* Token Description - Describes the signal structure;
* Functional Memory - Describes the standard technique for communication of functional information within the bounds of a performance model;
* Software Architecture Interface - Provides an overview of the approach to modeling software using the generic Honeywell processor performance model;
* Implementation Plan - Contains the plan for future VHDL performance model interoperability guidelines;
* Model Descriptions - Contains descriptions for primitive elements in the library, generic configuration capabilities, output capabilities, and RASSP DID requirements;
* Examples - Used to illustrates the application of VHDL performance models. Future releases of this document will contain more detailed examples of relevant RASSP architectures;

3. Model Development

The VHDL performance modeling methodology is targeted towards high level description, specification and performance analysis of computing systems. The tools and techniques themselves are not targeted towards any particular application. The level at which is appropriate to apply these tools is at the architectural level. Architectural level includes the actual device or entity under study such as a signal processor, and its environment, such as sensors and actuators. In the case of an electronics system, an architectural level description would include information about both the hardware and software. Note that the definition of "system" is loose here. While the application of performance models is constrained to electronic systems, the library should be fully capable of representing systems consisting of ASICs, boards, and subsystem cabinets, and sensor networks.

Regarding model capability, the two modeling/simulation areas which the existing HTC current performance models did not address well were large multiprocessor systems and signal and image processing application specific models. Since the application specific models are best addressed on a case by case basis, HTC is directing our activity to the multiprocessor modeling area.

The prior HTC modeling capability concentrated on small scale multiprocessing systems. As a result, the processor model developed into a powerful, highly flexible generic model. Communication models on the other hand have been limited to processor-memory bus models with rudimentary arbitration schemes.

The proposed modeling capability will include an efficient "light weight" processor models as well as a generic interprocessor communication models. Multiprocessor systems will require more elaborate communication models capable of more advanced protocols and arbitration mechanisms. Scalable point to point communication models capable of supporting several different protocols are needed to model large multiprocessor designs.

4. Related Contracts

Under a separate RASSP technology base contract, Omniview, in conjunction with HTC, will develop commercial quality product called the Performance Modeling Workbench (PMW). The PMW will provide an extensive graphical user interface for performance models. The performance models, based on the HTC VHDL performance model library, will adhere to the requirements specified in the interoperability guideline. The PMW will also include multi-processor modeling tools such as output analysis, capture, route/message cost, and architecture visualization.

Lastly, the performance models will be constructed in a manner to support hybrid modeling. Hybrid modeling supports performance analysis with interfaces to functional models of communication and other device models. Functional modeling of selected system components will be necessary. A good example of this is the detailed functional modeling of the underlying communication mechanisms. Under a RASSP technology base contract, Honeywell, along with the University of Virginia, will develop models and utilities to support hybrid modeling.

For more information on Honeywell Develops VHDL Performance Model Library, contact Fred Rose at rose@src.honeywell.com

Object-Oriented VHDL Provides New Modeling and Reuse Techniques for RASSP

by Dr. Sowmitri Swamy, Vista RASSP Program Manager

SCHAUMBURG, IL - Vista Technologies, Inc. and Martin Marietta identified new object-oriented constructs that can be implemented as an extension language to VHDL. This approach will improve system modeling and simulation, and increase the potential for component reuse.

The extension language, OO-VHDL, is a super-set of VHDL. It enables designers to mix OO-VHDL objects and traditional VHDL components as part of the same system description. It is important to note that this work is not defining a new language based on VHDL. It is identifying new constructs that can be implemented on top of VHDL.

A pre-processor implementation will generate simulatable VHDL code, which enables designers to leverage existing VHDL tools, such as analyzers and simulators. This approach provides users with the benefits of the language extension quickly and at modest cost, because no changes are needed in existing VHDL simulation tools. An object-oriented approach is already well established as the preferred method to designing complex systems, particularly software systems. Its benefits, based on several years of experience, have been widely documented. Until now, it has not been used to design complex systems that also involve hardware design, such as signal processing systems.

1. Improved System Modeling and Simulation

Extensive simulation is a key ingredient in the RASSP methodology to develop new signal processing architectures or to make model year upgrades to architectures. Simulations are performed at several levels - system, architecture, register-transfer - using models written in VHDL.

For commercial, off-the-shelf components, application-specific integrated circuits, and programmable logic, designers get simulation models from commercial model libraries or generate them from automated modeling tools. But at the higher levels of the design hierarchy, designers develop (abstract) models individually and manually during the design process. The burden of generating these models, and ensuring that they are fit for reuse in subsequent model years, means that the RASSP environment must provide support for quick model development with the capability for reuse.

System-level models contain abstract components that are eventually implemented as a hardware component, a software component, or a mixture of the two. The cornerstone of OO-VHDL is the Entity Object, which enables the object-oriented specification of an abstract component - whether it is a hardware design unit or a software object. Using the object-oriented concept of "inheritance," designers can reuse an EntityObject by making incremental changes in its behavior or interface.

The interaction between EntityObjects involves passing messages between them. These messages are commands to the recipient of the message to execute a specific operation. During system-level simulation, designers track the behavior of the system by examining the sequence of operations performed by various EntityObjects.

2. Elements of the OO-VHDL Approach

OO-VHDL provides new capabilities for encapsulation, reusability, inheritance, and message passing. Objects are typically described in two parts: an interface part and an implementation part, roughly corresponding to a VHDL entity declaration and architecture body. The object interfaces, known as class description, document the operations performed by an object. By reading a class description, it is easy for designers to determine the functionality of a component implementation part.

An object-oriented approach increases the potential for component reuse. For example, suppose a simple behavioral description of a highway/farm road traffic light controller exists, and you have to add left turn signals. The traditional approach would be to copy the old behavioral description and modify it. An object-oriented design reuse process factors out the common functionality of similar components in an inheritance hierarchy. Without inheritance, reuse can only occur at the component level; that is, either you must use the component as is, or you must design a new one.

Operations define an abstract procedural interface to an EntityObject. Messages invoke the operations. When an OO-VHDL message is sent, the invoker suspends operation until the corresponding action (the operation) is complete. From the sender's point of view, sending a message has the semantics of calling a procedure. This differs from the hardware model of communication through signals, which requires specialized protocols to synchronize behaviors and to exchange data between components.

However, unlike procedure calls, a message only requests that a particular operation be performed -- it does not cause the operation to execute immediately. An EntityObject services messages sequentially. To enforce sequentially, OO-VHDL queues all message requests sent if an EntityObject is actively servicing another operation request. When the current operation is completed, the next request is removed from the queue and serviced.

Concurrence control is an important aspect of system-level design. However, signal-based concurrence control, such as bus-resolution, is at too low a level. Of the existing concurrence control approaches, the distributed processing model (similar to the remote procedure call model) and the Ada rendezvous are the most desirable for high-level behavioral modeling because of their ease of use and generality. OO-VHDL combines the distributed processing and Ada approach; this combination is referred to as DP/A in OO-VHDL.

3. Modeling Examples Prove Effectiveness of Object-Oriented Approach

To explore and test new OO-VHDL language constructs, Vista developed example system-level models written in tandem with the development of the language extensions. OO-VHDL models for the IEEE 802.4 Token Passing Bus Access Method for standard LANs, and ESPS, an example signal processing system, proved the power of object-oriented language constructs. These models demonstrated the effectiveness of OO-VHDL for rapid prototyping of system behavior that involves hardware/software interactions, and complex synchronization of concurrent hardware or software entities.

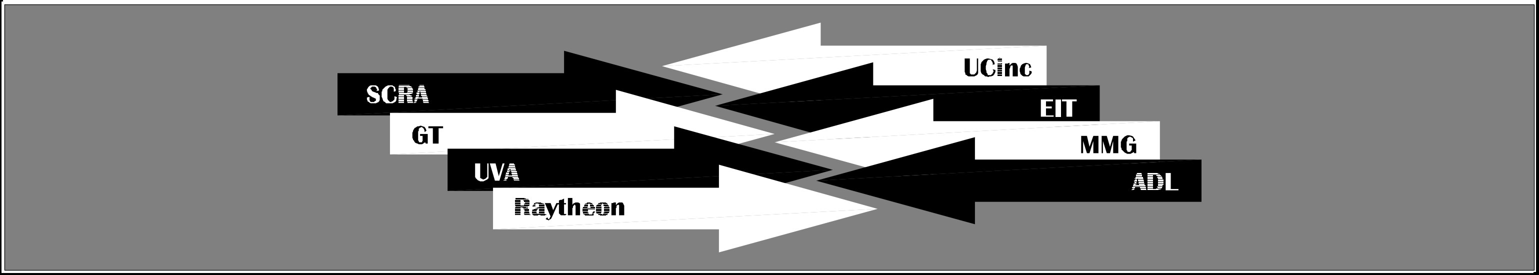
4. Language Standardization Effort

Standardization efforts are underway to add object-oriented features to VHDL in 1997 or sooner, if possible. An IEEE Design Automation Standards Committee group was formed for this purpose. The language and supporting tools will be made available free to users to generate valuable comments and suggestions.

5. The O-O VHDL Modeling Process

The pre-processor approach translates OO-VHDL models into VHDL 1076. It allows simulation tools used in the RASSP environment to be used for OO-VHDL simulation. This approach allows EntityObjects and VHDL components, including commercial, off-the-shelf components, to be freely mixed in any modeling situation. The OO-VHDL modeling process starts by generating the OO-VHDL models using a design entry tool. The modeler is provided with an extensive collection of OO-VHDL objects in an OO-VHDL design library that can be reused or extended in the object sense. The model is then converted into standard VHDL by the pre-processor and simulated. During simulation, the traceability tool allows users to track the simulation in terms of the original OO-VHDL model developed by the user.

For more information on modeling at the system and architectural levels, contact Dr. Sowmitri Swamy by phone at 708-706-9300, by fax at 708-706-9317, or by E-mail at swamy@vistatech.com.



The Ptolemy Kernel — Supporting Heterogeneous Design

by The Ptolemy Team

1. Introduction

A technology base team at the University of California at Berkeley is developing a software environment called *Ptolemy* that supports heterogeneous design. An early contribution of this effort has been the design of a compact software infrastructure upon which specialized design environments (called *domains*) can be built. The software infrastructure, called the *Ptolemy kernel*, is made up of a family of C++ class definitions. Domains are defined by creating new C++ classes derived from the base classes in the kernel.

Domains can operate in any (or all) of three modes:

Simulation -- A scheduler invokes code segments in an order appropriate to the model of computation.

Code generation -- Code segments are stitched together to produce one or more programs that implement the specified function.

Compilation -- The specification is analyzed and translated into optimized code in any target language.

At Berkeley, we have built a variety of domains that operate in the first two modes only, although code generation domains often have elements of optimization from the third.

The use of an object-oriented software technology permits each of these domains to interact with one another without knowledge of each others' features or semantics. Thus, using a variety of domains, a team of designers can model each subsystem of a complex, heterogeneous system in a manner that is natural and efficient for that subsystem.

2. The Design of the Kernel

The overall organization of the latest release of the Ptolemy system is shown in Figure 1. A typical use of Ptolemy involves starting two UNIXTM processes, as shown in Figure 1(a), the first containing the user interface (VEM) and the design database (OCT), and the other containing the Ptolemy kernel. An alternative is to run Ptolemy without the graphical user interface, as a single process, as shown in Figure 1(b). A textual interpreter called "ptcl" is used in this case. It is also possible to design other user interfaces for the system.

The executables "pigiRpc" or "ptcl" can be configured to include any subset of the available domains. The most recent picture of the domains that Berkeley has developed is shown in Figure 2. Many different styles of design are represented by these domains. More are constantly being developed both at Berkeley and elsewhere, to experiment with or support alternative styles.

The Ptolemy kernel provides the most extensive support for domains where a design is represented as a network of blocks, as shown in Figure 3. A base class in the kernel, called Block, represents an object in this network. Base classes are also provided for interconnecting blocks (PortHole) as well as for carrying data between blocks (Geodesic) and managing garbage collection efficiently (Plasma). Not all domains use these classes, but most current ones do, and hence can very effectively use this infrastructure.

Figure 3 shows some representative methods defined in these base classes. For example, note the initialize, run, and wrapup methods in the class Block. These provide an interface to whatever functionality the block provides, representing for example functions performed before, during, and after execution of the system.

Blocks can be hierarchical, as shown in Figure 4. The lowest level of the hierarchy, as far as Ptolemy is concerned, is derived from a kernel base class called "Star." A hierarchical block is a "Galaxy," and a top-level system representation is a "Universe."

3. Models of Computation

The Ptolemy kernel does not define any model of computation. In particular, although the Berkeley team has done quite a bit of work with dataflow domains in Ptolemy, every effort has been made to keep dataflow semantics out of the kernel. Thus, for example, a network of blocks could just as easily represent a finite-state machine, where each block represents a state. A particular domain defines these semantics. Suppose we wish to define a new domain, called XXX. We would define a set of C++ classes derived from kernel base classes to support this domain. These classes might be called "XXXStar," "XXXUniverse," etc., as shown in Figure 4.

The semantics of a domain are defined by classes that manage the execution of a specification. These classes could invoke a simulator, or could generate code, or could invoke a sophisticated compiler. The base class mechanisms to support this are shown in Figure 5. A "Target" is the top-level manager of the execution. Similar to a Block, it has methods called "setup," "run," and "wrapup." To define a simulation domain called "XXX," for example, one defines at least one object derived from Target that runs the simulation. As suggested by Figure 5, a Target can be quite sophisticated. It can, for example, partition a simulation for parallel execution, handing off the partitions to other Targets compatible with the domain.

A Target will typically perform its function via a Scheduler. The Scheduler defines the operational semantics of a domain by controlling the order of execution of functional modules. Sometimes, schedulers can be specialized. For instance, a subset of the dataflow model of computation called "synchronous dataflow" allows all scheduling to be done at compile time. The Ptolemy kernel supports such specialization by allowing nested domains, as shown in Figure 6. For example, the SDF domain

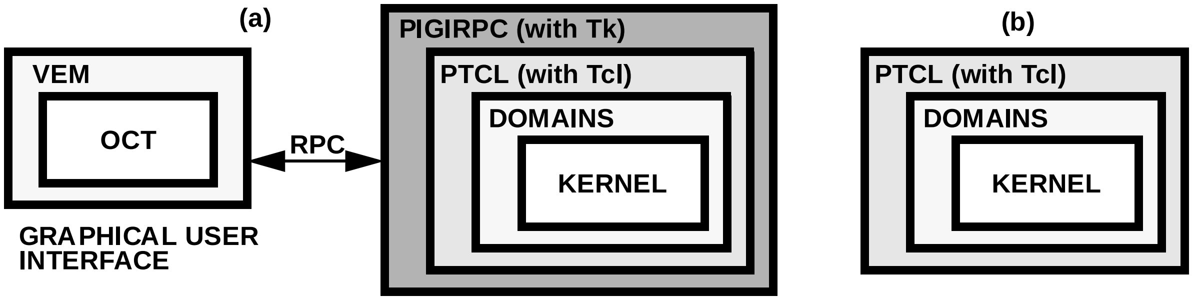


Figure 1. The overall organization of Ptolemy version 0.5.1, showing two possible execution styles. This report concentrates on the *kernel* and its relationship to the *domains*.

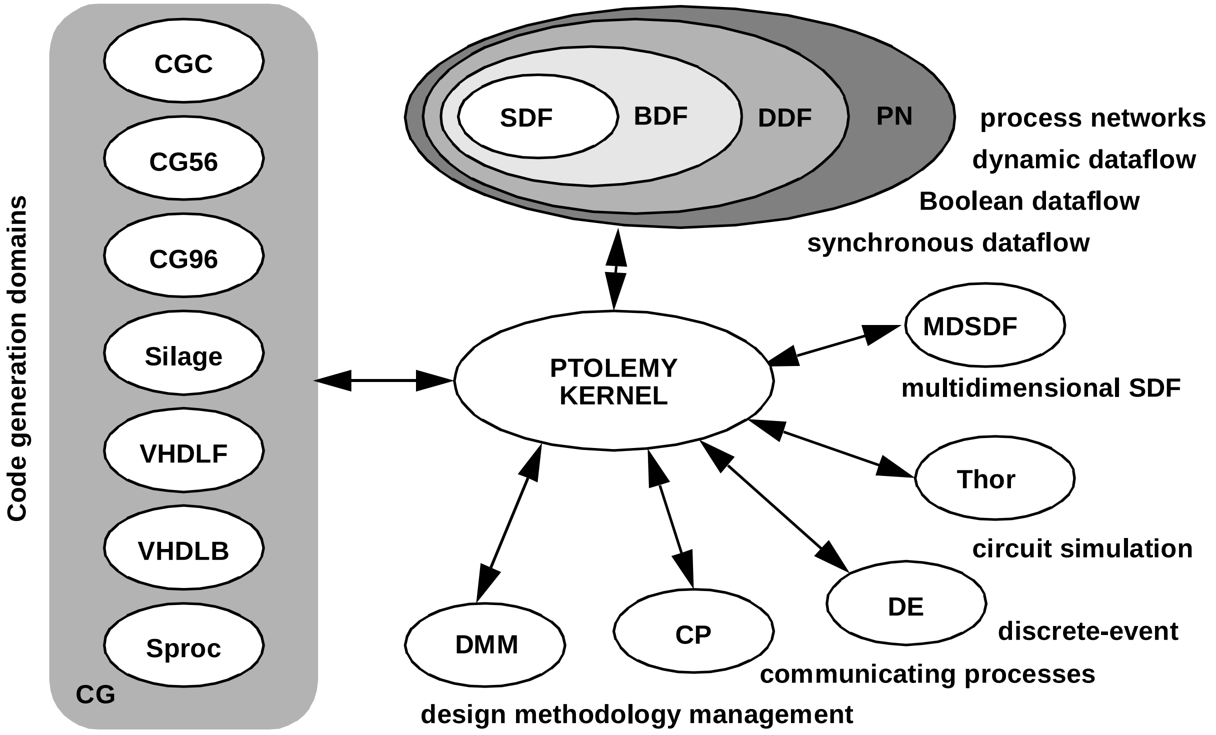


Figure 2. The most recent view of the set of domains developed at Berkeley. This article will discuss only CG, which underlies all of code generation.

(see Figure 2) is a sub-domain of the BDF domain. Thus, a scheduler in the BDF domain can handle all stars in the SDF domain, but a scheduler in the SDF domain may not be able to handle stars in the BDF domain. A domain may have more than one scheduler.

4. Mixing Models of Computation

Domains in Ptolemy can be mixed. Thus, one system-level design can contain multiple subsystems that are designed or specified using different styles. The kernel support for this is shown in Figure 7. An object called "XXXWormhole" in the "XXX" domain is derived from "XXXStar," so that from the outside it looks just like a primitive in the XXX domain. Thus, the schedulers and targets of the XXX domain can handle it just as they would any other primitive block. However, inside, hidden from the XXX domain, is another complete subsystem defined in another domain, say "YYY." That domain gets invoked through the setup, run, and wrapup methods of XXXWormhole. Thus, the wormhole is polymorphic in a broad sense.

5. Code Generation

The domains shown in Figure 2 are divided into two classes: simulation and code generation. In the simulation domains, a scheduler invokes the run methods of the blocks in a system specification, and those methods perform some function associated with the design. In code generation domains, the scheduler also invokes the run methods of the constituent blocks, but these run methods synthesize code in some language. i.e., they

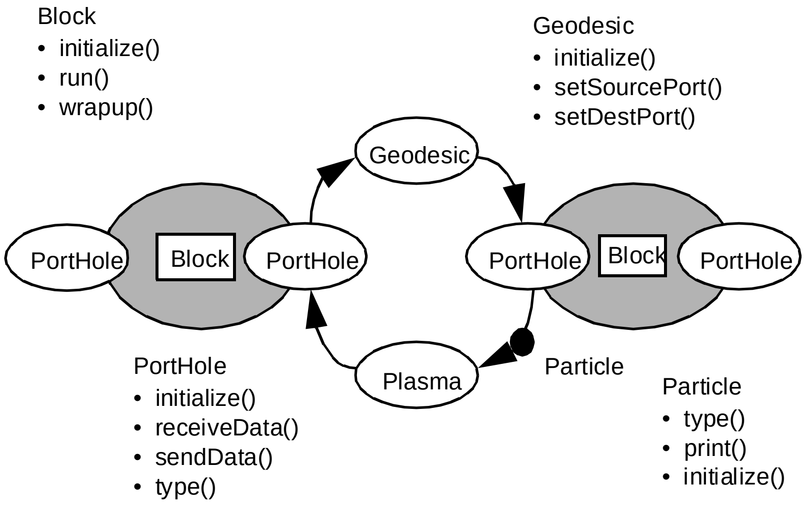


Figure 3. Block objects in Ptolemy can send and receive data encapsulated in Particles through Portholes. Buffering and transport is handled by the Geodesic and garbage collection by the Plasma. Some methods are shown.

generate code to perform some function, rather than performing the function directly. The Target is responsible then for generating the connecting code between blocks (if any is needed). This mechanism is very simple, and language independent. We have built code generators for a number of languages, as indicated in Figure 2.

An alternative mechanism that is supported but less exploited in current Ptolemy domains is for the target to analyze the network of blocks in a system specification and generate a single monolithic implementation. This is what we call compilation. In this case, the primitive blocks (Stars) must have functionality that is recognized by the target. In the previous code generation mechanism, the functionality of the blocks is arbitrary and can be defined by the end user.

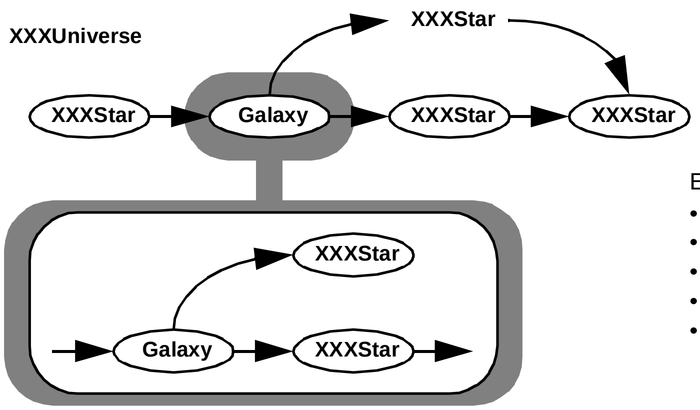
6. Conclusions

In summary, the key idea in the Ptolemy project is to mix models of computation, implementation languages, and design styles, rather than trying to develop one, an all encompassing technique. The rationale is that specialized design techniques are (1) more useful to the system-level designer, and (2) more amenable to high-quality high-level synthesis of hardware and software. The Ptolemy kernel demonstrates one way to mix tools that have fundamentally different semantics, and provides a laboratory for experimenting with such mixtures.

The Ptolemy kernel has been used successfully outside Berkeley for a number of domain designs. A notable example is the work of Berkeley Design Technology, Inc., as part of the Martin Marietta RASSP program, to use the Ptolemy to connect the SPW and Bones tools from the Alta Group at Cadence.

More information about the Ptolemy project, plus access to all of the software and documentation, is available on the worldwide web via the URL http://ptolemy.eecs.berkeley.edu.

The current Ptolemy team is: Shuvra Bhattacharyya, Joseph T. Buck, Wan-Teh Chang, Brian L. Evans, Steve X. Gu, Sangjin Hong, Christopher Hylands, Asawaree Kalavade, Alan Kamas, Allen Lao, Bilung Lee, Edward A. Lee, David G. Messerschmitt, Praveen K. Murthy, Thomas M. Parks, José Luis Pino, Farhana Shiekh, S. Sriram, Juergen Teich, Warren W. Tsai, Patrick J. Warner, Michael C. Williamson.



Examples of Derived Classes
- class Star:: Block
- class XXXStar:: Star
- class Galaxy:: Block
- class Universe:: Galaxy, Runable
- class XXXUniverse:: Universe

Figure 4. A complete Ptolemy application (a Universe) consists of a network of Blocks. Blocks may be Stars (atomic) or Galaxies (composite). The “XXX” prefix symbolizes a particular domain (or model of computation).

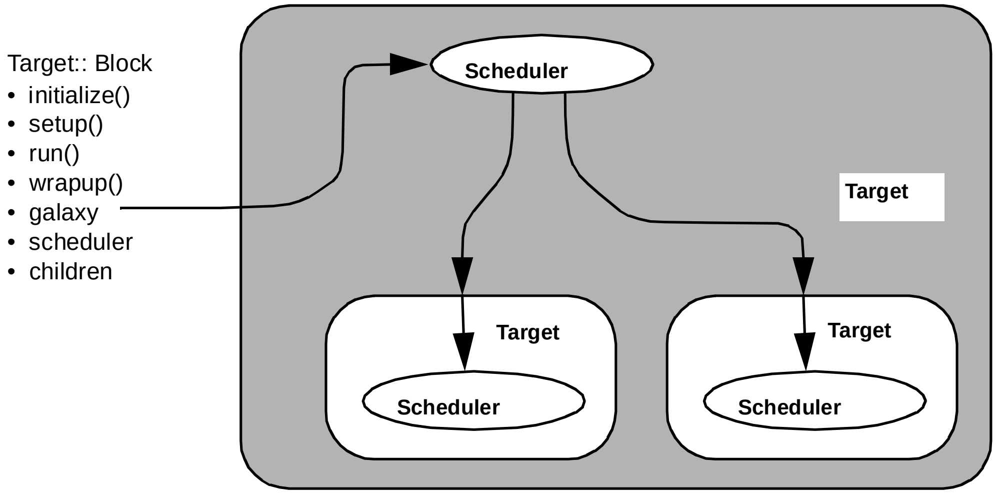


Figure 5. A Target, derived from Block, manages a simulation or synthesis execution. It can invoke it's own Scheduler on a Galaxy, which can in turn invoke Schedulers in sub-

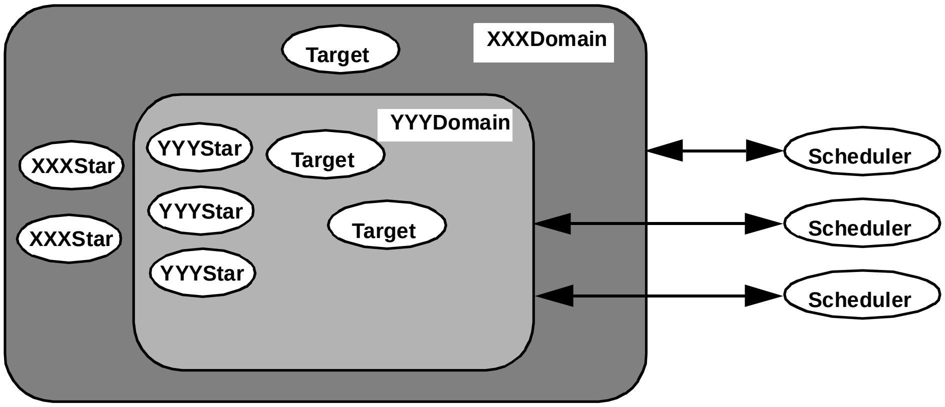


Figure 6. A Domain (XXX) consists of a set of Stars, Targets and Schedulers that support a particular model of computation. A sub-Domain (YYY) may support a more specialized model of computation.

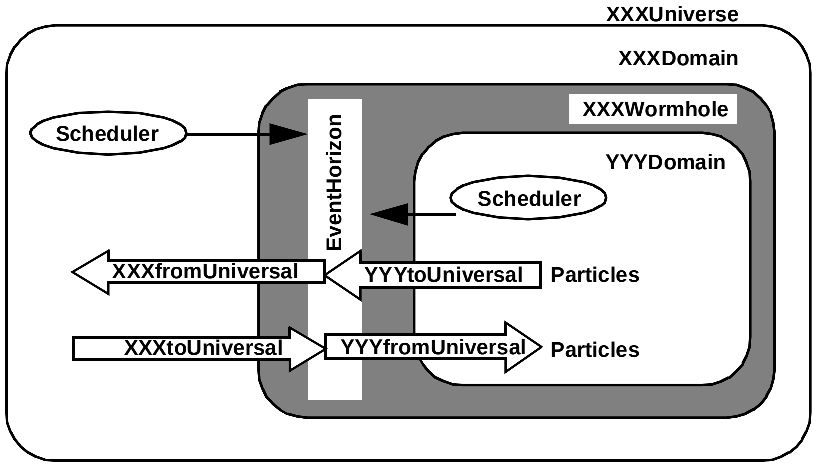


Figure 7. The universal EventHorizon provides an interface between the external and internal domains.

VHDL Component Modeling: Impact on the RASSP Program

by J. Scott Calhoun and Dr. Bob Reese

Abstract

Mississippi State University's Microsystems Prototyping Laboratory (MPL) has been contracted as part of the RASSP Technology Base to develop VHDL models of selected Cypress Semiconductor standard integrated circuits. Mississippi State and Cypress have entered an agreement whereby Cypress will provide timing information necessary to create VHDL models of high quality and accuracy to be released as part of the RASSP program. MPL is also under contract to delivery VHDL-based tools which will assist the model developer in testing created VHDL models. Finally, MPL will be participating in the RASSP E&F through the development and release of a HTML-based VHDL course (this effort is currently nonfunded).

1. Component VHDL Models and the RASSP Design Process

The Rapid Prototyping of Application Specific Signal Processors (RASSP) initiative is intended to dramatically improve the process by which complex digital systems, particularly embedded digital signal processors, are designed, manufactured, upgraded, and supported. RASSP seeks an improvement of at least a factor of four in the time required to take a design from concept to fielded prototype or to upgrade an existing design, with similar improvements in design quality and life cycle cost. The motivation for RASSP is the need to provide affordable embedded signal processors for a wide range of DoD systems that are state-of-the-art when they are fielded, rather than when they are first defined.

To accomplish these goals, improvements are being sought at all levels of today's system design methodology. In addition, RASSP is pushing the envelope upward in the electronics design process to develop tools and capability which will integrate system requirements and specification development; along with top-level hardware and software design development into the overall RASSP design flow seamlessly. The primary tangible output from a successful iteration of the RASSP design process will be the ability to produce virtual or model-year prototypes of complex digital electronic systems. These virtual prototypes will contain all the information necessary to quickly and successfully manufacture physical prototypes which will meet the documented system requirements. Simulation of the model-year prototypes utilizing VHDL (or some derivative there of) is the main verification mechanism by which the system is shown to meet the requirements levied against it.

At the design data level where individual system printed circuit boards are to be produced, simulations exist where there is a one-to-one correspondence between inner connected simulation models and the physical representation of the integrated circuits. From a system perspective this is the last and lowest level of simulation performed to verify the design correctness prior to fabrication. Many of the integrated circuits used in a PCB design may be ASICs. Those that are not fall into a broad category of circuits which are referred to as standard off-the-shelf components. This class of components include microprocessors (general purpose, signal, FPU, etc.), bus interface components (VME, MIL-STD-1553, etc), memory (RAM, PROM, FIFO, etc.), programmable logic devices (EPLD, PAL, FPGA, etc.), and specialized bus drivers (buffers, latches, transceivers). These components represent the bulk of the digital standard components sold by semiconductor companies in the world today for use in complex digital system designs. The RASSP Technology Base program has funded several VHDL modeling efforts to assist the RASSP program in obtaining VHDL models necessary to develop and demonstrate the overall RASSP design process. Mississippi State University has been funded to develop VHDL models of memory and programmable logic device standard components offered by Cypress Semiconductor, Inc. When completed, these models (along with others developed by the RASSP program) will be used to create the board level simulations necessary to verify correctness of RASSP designs prior to fabrication. This verification allows for virtual board designs to be debugged in simulation avoiding costly fabrication rework cycles.

2. VHDL Modeling at MSU

MSU has been tasked to provide VHDL models for standard Off-The-Shelf (OTS) logic devices provided by Cypress Semiconductor, Inc. Cypress has a design division located in Starkville where many of the components targeted for modeling ware designed. This affords ready access to internal data which may be needed to insure the accuracy and performance of developed models. Cypress has entered an agreement to work with MSU to provide all component information necessary to deliver to the RASSP program the highest quality of models possible.

The program calls for MSU to address the following part types:

1. Flash PLD
2. FPGA
3. Erasable PLD
4. Dual Port SRAM
5. Single Port SRAM
6. PROM
7. FIFO

The modeling strategy developed at MSU will be to develop baseline methodologies for each part type listed above. In addition, as part type methodologies are developed, part models for multiple components in a part type will be developed as quickly as possible to deliver the maximum number of models possible. The following subsections describe the development strategy for each of the part types which are currently under development.

2.1 Single Port SRAM and PROM

Two general VHDL packages have been developed for generic memory device modeling. Both packages are memory-organization independent.

The packages are as follows:

1. A general memory package which statically allocates all model storage. This is suitable for ROM models and small RAM models.

2. A general memory package which dynamically allocates model storage and has a user-controlled option for swapping memory pages to disk. This is suitable for large and small RAM models. The models developed to date are: 32K x 8 EPROM (CY7C256), 32K x 8 SRAM (CY7C199- CMOS, CY7B199 - BiCMOS), 64 x 4 SRAM (CY7C195 - CMOS, CY7B195 - BiCMOS). The EPROM model uses the static memory model while the SRAM models use the dynamic model. The SRAM models utilize approximately 98% common code with timing packages accounting for most of the non-shared code.

2.2 Flash and Erasable PLD

A 22V10 PLD model (PALC22V10d) has been developed; the model makes extensive use of GENERATE statements based upon the JEDEC map to create the internal simulation structure. The GENERATE methodology gives good runtime performance in that only the programmed portions of the device contribute to the simulation overhead. We plan on following this methodology in modeling the more complex Cypress PLDs represented by the CY7C37X family.

2.3 Dual Port SRAM

We are currently studying the issue of dual port SRAM modeling. Dual port SRAMs such as the CY7B138 with on-board arbitration offer significant modeling challenge.

2.4 FPGA

Our FPGA modeling plans center around the Cypress pASIC380 family. At this time we have held only preliminary discussions with Cypress concerning possible modeling approaches.

3. VHDL Component Modeling Issues

The amount of VHDL component models available today is still relatively small. Therefore, there are and will be a number of modeling issues with regard to VHDL component models as models become available to the RASSP community as part of this program. These issues are briefly discussed.

3.1 EIA-567A

EIA-567A defines three packages and a recommended methodology for representing timing, electrical and physical view information in VHDL. At this time the EIA-567A specification has been followed in those areas in which it was felt that EIA-567A added value. To date, the timing view package has been utilized to a limited extent. Examples of 567A timing view compliance include adding generics for input wire delay and output load delay and generics for controlling 'X'-value and message generation on timing violations. MSU did not follow the recommended methodology for encoding timing parameters; a specialized timing package for representing databook timing values was developed. One reason for not using the 567A methodology is that it requires renaming the databook timing parameters to follow a generic template. This puts the burden on the model writer to deal with timing parameter name translation which can be error-prone. This also makes the model source code more obtuse to external readers who know the databook timing parameters and are not familiar with the 567A specification. Finally, there is no provision in the EIA-567A timing view for having different speed grades (e.g. -10, -15, -20), short of writing a different package for each grade which makes model code maintenance and test bench configuration awkward.

3.2 Model Portability and Interoperability

MSU has tested the models in both Vantage and Model Tech environments in order to ensure portability of the models. As per EIA-567A, the models utilize the IEEE-1164 standard logic package for the state/strength value system of the models. MSU is in the process of obtaining a set of models which will act as an interoperability test set. These models include a processor and several component models. MSU plans to create testbenches where MSU developed models and the obtained models will simulate in the same environment to ensure interoperability of MSU models with other industry developed models. In addition, MSU plans to work with other RASSP Technology Base contractors to create VHDL simulations which demonstrate the viability of VHDL as a simulation media throughout the RASSP design process.

4. Model Documentation and Release

VHDL component models developed by MSU are being documented and released via the World Wide Web (WWW). MSU has obtained permission from Cypress to post Web versions of the three component datasheets representing part of the initial model release. The model release mechanism will be built into the electronic datasheet for each part. Because of export restrictions on RASSP deliverable, VHDL model release will be limited to RASSP program participants.

4.1 WWW Component Datasheets

The WWW component datasheets are viewable from the MSU RASSP homepage **(http://www.erc.msstate.edu/mpl/rassp/html/overview.html)** by clicking on the VHDL Model Library icon. This presents the overall Cypress BiCMOS/CMOS Databook from which the datasheets for the parts to be modeled are contained. Traversal of the databook homepage reveals sections for part types (STATIC RAMS, PROMS, EPLDS). Each listing in these part types document represents a part that is either modeled or is projected to be modeled. Those which are complete are hyperlinked to the homepage for the individual circuit. A portion of the homepage for the 27C256 is shown on the next page (see Figure 1.)

4.2 WWW Component Model Release

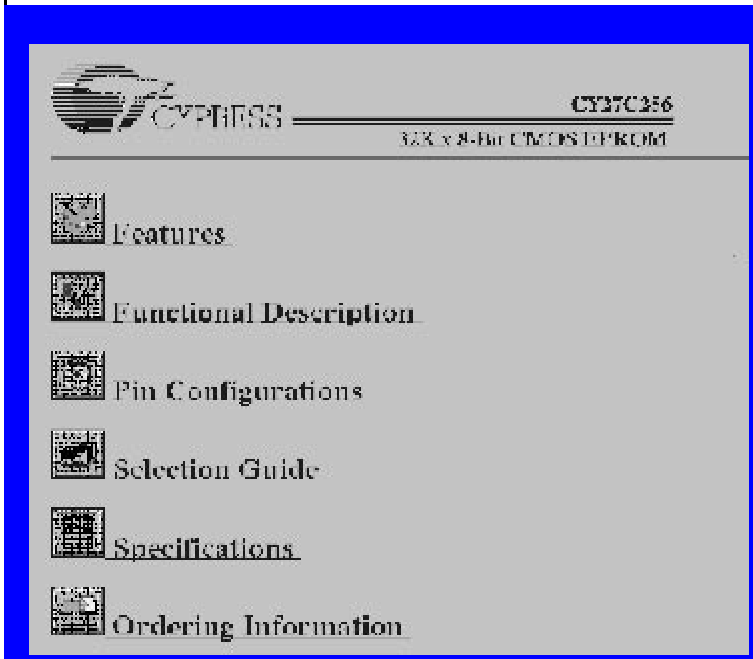
RASSP VHDL models are export restricted. For initial releases from the RASSP Technology Base programs, the models are being released via the WWW utilizing restricted access and data encryption. There will be a single point of contact for model release per RASSP contractor or government organization. The procedure for release RASSP VHDL models is as follows:

1. Model release is an extension of the WWW component datasheet.
2. Model release portion of the WWW component datasheet is http access protected with user/password required for access.
3. Data encryption (des) is utilized with user key assigned by MSU.

4. User/passwd/deskey authentication required.
   - o single point RASSP contractor release
   - o phone call authentication required
   - o user/passwd/deskey assign over phone
5. WWW user/passwd used to access release for in component datasheet.
   - o single file for each model
   - o des encryption prior to transfer
   - o ftp transfer to user site
   - o des decrypting by user with deskey
   - o each transfer logged

An example model release is shown below:

> *For more information on the VHDL Component Modeling contact J. Scott Calhoun at jscott@erc.msstate.edu*



CYPRESS CY27C256
32K x 8-Bit CMOS EPROM

Features

Functional Description

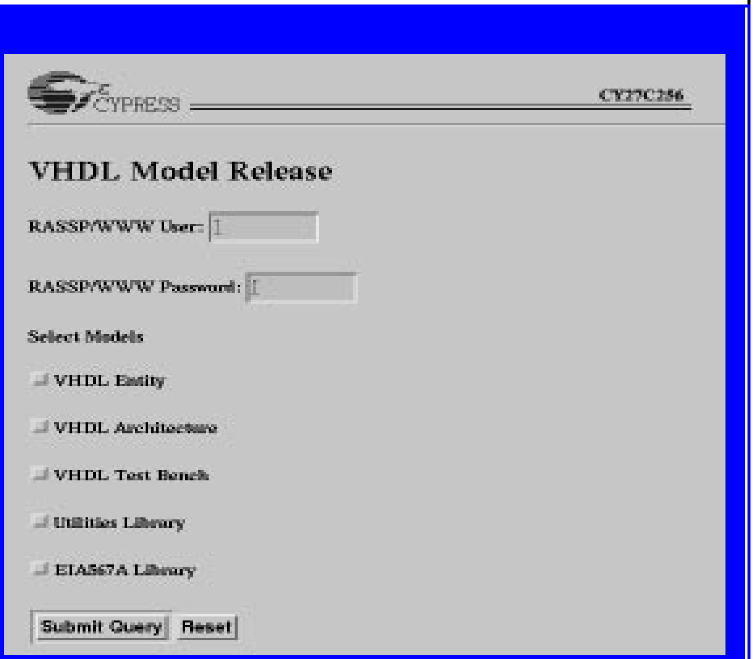
Pin Configurations

Selection Guide

Specifications

Ordering Information

Figure 1. 27C256 WWW Component Datasheet



CYPRESS CY27C256

VHDL Model Release

RASSP/WWW User:

RASSP/WWW Password:

Select Models

□ VHDL Entity

□ VHDL Architecture

□ VHDL Test Bench

□ Utilities Library

□ EIA567A Library

Figure 2. 27C256 VHDL Model Release Form

Assessing and Improving Current Practice in the Design Of Application-Specific Signal Processors

by G. A. Shaw and V. K. Madisetti

Abstract

The Department of Defense ARPA program for Rapid Prototyping of Application Specific Signal Processors (RASSP) exists to significantly improve the process by which embedded digital signal processors are developed (prototyped) and supported (maintained and upgraded). As used in the RASSP program, the term prototype signifies a system that is a precursor to a deployed system, but still meets all of the essential performance goals and is designed to facilitate maintainability and upgradeability. In this paper, current practice in the design of embedded digital signal processors, as exemplified in the traditional waterfall design methodology, is examined and shortfalls in the design methodology and supporting tools are identified. Opportunities for improving the traditional design practice are then identified and evaluated in terms of potential benefits, as well as impediments, to implementation and adoption by the community.

1. Introduction

Within the past 10 years, high-end signal processing applications have grown from millions of operations per second, implemented in hard wired or uni-processor architectures, to billions of operations per second implemented on arrays of programmable multiprocessors. At the same time, the functionality that was once implemented at the board level with large-scale integrated circuits has been subsumed at the chip level in very large scale integrated circuits containing millions of transistors and hundreds of pins, employing clock rates approaching 100 MHz and complex software development environments. The introduction of more complicated building blocks, higher clock rates, and tightly-coupled hardware and software environments has opened a gap between traditional design and verification methods and the complexity and supportability required for contemporary digital signal processors. Furthermore, with new DSP chip technology being introduced annually, traditional methods of optimizing a design for a given application and associated processing engine are no longer cost-effective or appropriate in terms of supporting an upgrade path.

Applications requiring embedded signal processors are as numerous and diverse as the methodologies employed to design and build them. Consequently, there is no unique model of “current practice” as it applies to the design of application specific signal processors. Nevertheless, a representative model of current practice is essential to the RASSP program in order to assess where improvements are needed, and also as a basis for measuring progress of the RASSP program toward the goals of reduced design cycle time, reduced cost, and improved quality.

The focus of the RASSP program is on high-performance form-factor constrained signal processors consisting of anywhere from a few to hundreds of processing engines. The models described here are an attempt to characterize, at least in an average sense, the current practice in developing such state-of-the-art embedded signal processors at the inception of the RASSP program, circa 1993.

2. Current Practice Model Views

In developing a representative or ‘‘industry standard’’ model of current practice, there are at least three views of interest to the RASSP program which are shown in Table 1.

Table1: Current practice Views

VIEW	EXAMPLES
Process Design Flow	Productivity, Test, Reviews
Resource Development Time	Cost, Tools, Libraries
Product Form	Performance, Defects, “...ilities”

The process view emphasizes issues such as the steps or methodology followed, the degree of concurrent activity, and the productivity achieved. The resource view might also be termed generalized cost, and emphasizes the people, tools, time, etc. required to develop a prototype signal processor. The product view emphasizes issues related to the soundness and performance of the product, as well as adherence to requirements. The three views are clearly not independent. For example, methodology affects development cost (resource) and quality (product).

2.1 Process View

The traditional design methodology, which is embodied in military process standards such as DOD-STD-2167A for software development, is a waterfall design process, as illustrated in Figure 1.

The underlying concept behind the waterfall design process is a progression through various levels of abstraction, or phases, with the intent of fully characterizing each level of abstraction before moving to the next level, and providing a comprehensive work package at each phase. Strict adherence to the waterfall design methodology is impractical, in part because the requirements for an embedded signal processor are often vague at the beginning of a project, and the processor is often subject to significant design changes. For example, in a radar system, waveforms, processing algorithms, and subsystem interfaces may all be modified during the course of signal processor development. Nonetheless, this methodology is characteristic of current practice, particularly in the defense industry. While following the waterfall design methodology does not preclude attaining the RASSP goals of rapid design cycle time, low life cycle cost and model year upgrade capability, the waterfall design methodology does tend to foster a number of bad design practices including:

1. Low exploitation of concurrent engineering,
2. Emphasis on wrong problems early in the design phase
3. Inflexibility late in the design phase,
4. Low level of customer interaction and subsequent satisfaction,
5. Significant rework and cost resulting from not discovering design flaws until the integration phase.

Figure 2 is a simplified representation of a waterfall design methodology for embedded signal processor design. Perhaps the most sig-

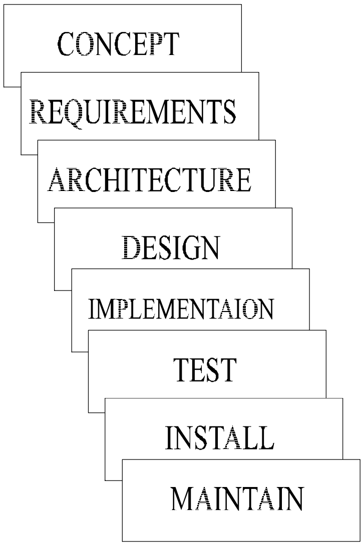


Figure 1. Waterfall development methodology.

Figure 2. Simplified current practice design flow.

nificant deficiency in the methodology is that hardware subsystems and application software are not integrated until late in the process, and significant design flaws may go undetected until the integration step.

2.2 Resource View

Figure 3 is an estimate of the relative distribution of cost associated with the development of the synthetic aperture radar (SAR) image formation processor described in [1]. A number of activities such as program management and reporting have been omitted from Figure-3 for simplicity. The performance requirements for the processor are summarized in Table 2.

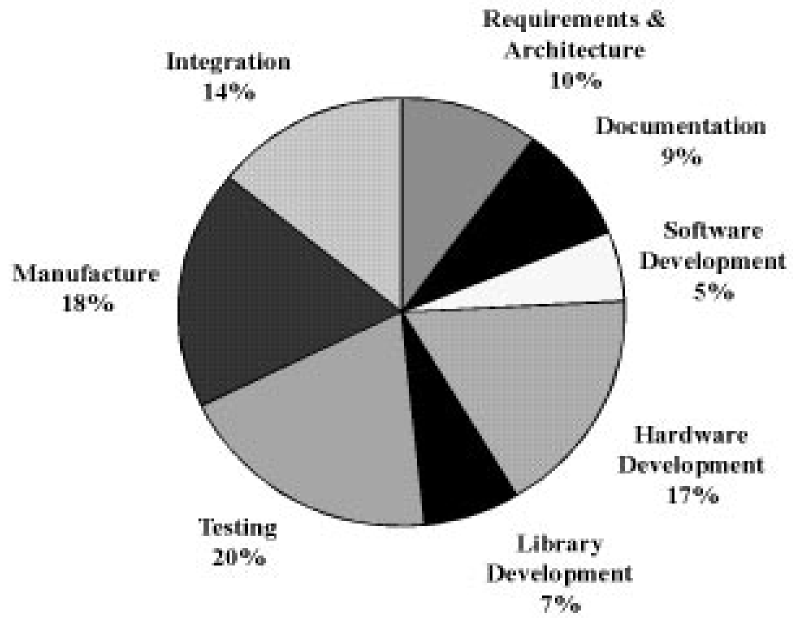


Figure 3. Relative dollar cost associated with various development tasks.

Table2: SAR Processor Requirements

ITEM	REQUIREMENT
Max. Volume	2.2 cuf
Max. Power	500 W
Max. Weight	60 lbs
I/O Rate	18/27 MB/s
Interface	Fiber
Polarizations	3
Frame Size	2048x512 pixels
Dynamic Range	>103dB

In Figure 3, requirements and architecture development together represent only 10% of the total. However, once an architecture is selected, much of the development and life cycle cost of the system, as well as achievable performance, are determined. Also note that testing and integration consume a larger percentage of cost than the combined software and hardware development (34% versus 22%). The software development in this example consists mainly of well-defined algorithms such as FFTs.

State-of-the-art electronic design automation (EDA) tools to support the design flow of Figure 2 may cost as much as $80K-$100K per single-user license, and a collection of these tools, spanning the end-to-end process, could cost in excess of $1M to purchase, and $150K or more a year to maintain, excluding training costs. Despite the cost of the tools, interoperability across tools is not assured, particularly in the case of commonly used high-level system design tools. Computer-aided design (CAD) and computer-aided software engineering (CASE) tools are available to support either the hardware or software design, but there is little to support the co-design and simulation of hardware and software. In particular, there are few libraries and models to support co-simulation of hardware and software, and there are not standards for interoperability of models at various levels of design abstraction.

3. Opportunities For Improvement

Incremental improvements in design practice occur more or less continually, but significant improvements are almost always due to a revolutionary change in the resources or processes employed [2]. The shortfalls in traditional design methodology suggest areas which might be targeted for revolutionary change.

3.1 Opportunity for Process Improvement

3.1.1 Executable Requirements

Figure 2 emphasizes the fact that current practice is to provide processor requirements in written form, often hundreds of pages of requirements which must be interpreted and captured in a traceability tool. Provision of requirements in machine readable and executable form has the potential to significantly reduce the ambiguity in written requirements. The SAR benchmark described in [1] includes an executable requirement written in VHDL which is intended to serve as the basis for test bench generation during detailed design and verification.

3.1.2 Virtual Prototyping

A virtual prototype [3], consisting of a software model of the hardware executing a representation of the application code, has the potential to uncover design flaws before the costly step of hardware fabrication and test fixture generation. In the case of integrated circuit design, technology for first-pass correct design. The same concepts can be applied to board level design provided suitable models and simulation tools are available. A virtual prototype also has the potential to support early customer evaluation and exploration of performance trades.

3.1.3 Successive Refinement

Unlike the waterfall development methodology, which emphasizes complete descriptions of the signal processor at each level of abstraction in the design process, successive refinement emphasizes rapid development of a less than full function prototype to uncover potential problems early and to influence the design through hands-on experience. The terms successive refinement, spiral design, incremental development, risk-driven design are all used somewhat interchangeably to describe this basic approach. Spiral design was pioneered for software development [4], but the concept can be applied to hardware as well. Benefits of successive refinement include the ability to involve the end user in evaluating early prototypes, early discovery of problems in the design concept, and improved estimates of the cost and schedule to produce a fully-functional prototype. However, successive refinement can be costly when hardware fabrication is in the loop, and virtual prototyping represents a potentially cost-effective methodology for supporting successive refinement.

3.1.4 Co-development Methodologies

Co-development, or hardware-software co-design, refers to the ability to begin with an implementation-independent representation of the requirements for a signal processor and evolve these requirements to a hardware and software implementation that is optimum, or nearly so, in some sense. Virtual prototyping supports hardware-software co-design by facilitating the transfer of functionality between hardware and software, enabling performance analysis and trade-offs prior to the existence of the hardware. Current practice predominantly relies on the experience of designers to allocate functionality to either hardware or software early in the design. Once the allocation is made, the hardware and software development tends to proceed along relatively independent paths with few opportunities created for improvement through trade-off analyses.

3.1.5 Parametric Cost Estimation

Parametric cost estimators (PCEs) have been shown to give engineering managers a competitive edge by accurately predicting project costs. PCE tools, available now in stand-alone form, can be integrated with front-end design tools to provide a more quantitative and traceable basis for architecture selection. In the absence of a well-defined cost estimation methodology, critical items, such as testability, are often overlooked in determining cost, schedule and risk associated with candidate architectures. Representing each candidate architecture by a set of cost breakdown structures and applying the appropriate PCE tools helps ensure that all relevant aspects are considered. PCE tools also enable assignment of numerical values for cost, schedule and risk associated with each candidate architecture and provide documentation of the basis for architecture selection.

The ability to identify and quantify life cycle cost issues is an important capability afforded by PCE tools. Hardware life cycle costs are a function of maintenance concept (e.g. throw away vs. fix a failed module), and a specific maintenance concept must be supported by the appropriate built-in test features and external test equipment. PCE tools provide a means for rapidly evaluating a large number of maintenance concepts, and results of the PCE life cycle analysis have a direct impact on test requirements and architecture selection.

3.2 Resource Improvements

3.2.1 Standard Hardware Interfaces

As DSP chips continue to gain in complexity and functionality, the major effort in embedded processor design has shifted to specialized hardware for systolic processing and the communication and control interfaces for multiprocessor architectures. The VME bus is a familiar example of a standard interface which facilitates rapid development of application hardware by promoting reuse and standard protocols for communication. However, bus architectures do not scale well, and interfaces with substantially higher bandwidth and latency are required for many applications. Designing for upgradeability demands the use of standard, scalable interfaces and memory architectures. Standard interfaces are essential in promoting widespread software and hardware reuse.

3.2.2 Reuse Libraries

DSP chip developers currently provide C-language instruction set simulators and highly optimized FFT and other software modules with a new chip. The provision of these tools and libraries promotes reuse on a wide scale. However, virtual prototyping and hardware-software co-development methodologies require many additional models at at various levels of abstraction. Currently, such models are not widely available, and concerns exist over the intellectual property embodied in such models. Modeling standards including, for example, the appropriate levels of abstraction, are needed to support wide-spread reuse.

In the case of application software, substantial reuse has been shown to yield gains of or more in productivity for uni-processor development [5]. However, substantial reuse of software in embedded signal processing is hampered by the lack of standard communication and control interfaces and the highly parallel hardware. Reuse can be enhanced by the architecture concepts described below.

3.3 Product Improvements

3.3.1 Model-Year Architectures

Programmable processing chips tend to double in performance approximately every 18 months, and, with multiple vendors developing new chips, improved technology is available on even shorter cycles. In order to field signal processing systems with the latest available processor technology, the hardware and software architectures must be sufficiently “portable” or standardized to support late binding of the processor chips to the software and board level communication fabric. In the case of software, this flexibility is achieved through high-level language implementation for the control and standardized library calls for DSP number crunching, such as FFTs. In the case of the hardware, the equivalent of a high-order language is a high-level description of the custom designs which can be synthesized into a preferred technology, such as FPGAs. The equivalent of the optimized DSP library is standardized interfaces and associated communication protocols.

3.3.2 Executable Specifications

In the same way that executable requirements facilitate the initial development of a signal processor, the final design can also be documented in a machine readable and executable form. Executable specifications have long been the norm for application software written in a high-level, portable language. The existence of standards for hardware design languages affords the opportunity to document hardware in a similar fashion, facilitating upgrades and reducing life-cycle support costs.

4. Conclusion

Historically, significant improvements in the required design-cycle-time and cost to produce embedded digital signal processors have been brought about by revolutionary changes in the design process or resources comprising the design environment. Presently, the process from schematic entry to printed wiring board, or from CASE tool to application code, is fairly mature. However, substantial improvements are feasible in the front-end processes relating to requirements capture, functional modeling, partitioning into hardware and software, and designing for easy upgradeability and supportability.

A MOSAIC server has been established on the World Wide Web as a source of additional information and publications. The Lincoln Laboratory RASSP home page is accessible via the uniform resource locator (URL) http://rassp.scra.org.

References

[1] B. W. Zuerndorfer, et al, ‘‘RASSP Benchmark-1 Technical Description,’’ MIT Lincoln Laboratory Project Report RASSP-1, 13 December 1994.

[2] K. A. Radtke, ‘‘The AT&T Hardware Design Environment: A Large System’s Hardware Design Process,’’ 31 ACM/IEEE Design Automation Conference, 1994.

[3] V. K. Madisetti, et al, ‘‘Virtual Prototyping of Embedded DSP Systems,’’ Proceedings IEEE International Conference Acoustics, Speech, and Signal Processing, 1995.

[4] B. W. Boehm, ‘‘A Spiral Model of Software Development and Enhancement,’’ ACM Software Engineering Notes, August 1986.

[5] R. B. Grady Practical Software Metrics for Project Management and Process Improvement, Prentice Hall, NJ, 1992.

For more information on the Assessing and Improving Current Practice in the Design of Application-Specific signal Processors contact vijay.madisetti@ee.gatech.edu or shaw@ll.mit.edu

Editorial Viewpoint

by Anthony J. Gadient

One of the most important factors contributing to the dramatic reduction in development time provided by the RASSP program is the effective use of the Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL). For this reason, we have chosen to focus on the RASSP VHDL activities in this edition of the RASSP Newsletter.

One of the activities being undertaken by RASSP which will contribute significantly to the effective use of VHDL in top-down system-level design is the development of a "taxonomy" of VHDL models. Many of you might wonder what the benefit of this activity will be and why it is being undertaken. This editorial is aimed at addressing this question by explaining the benefits a formal taxonomy of VHDL models will provide to the RASSP and VHDL user communities.

The RASSP design methodology and a VHDL taxonomy are like the ‘yin’ and the ‘yang,’ highly interrelated concepts, each inseparable from the other. To understand the importance of one, it is necessary to understand the other. Therefore, to explain the utility of a VHDL taxonomy it is first necessary to examine what a design methodology is.

A design methodology may be characterized as a directed, cyclic graph where cycles represent the iterative define-analyze-refine process that distinguishes design from other activities. For this reason, many work flow management systems use directed graphs as a way to present work flows (or design methodologies) to the end-user. Given this characterization of a design methodology, one can think of the nodes of the graph as representing design activities which often involve the invocation of design tools. Edges may be thought of as representing the flow of information and control from one design activity to the next. It is here that the idea of a VHDL taxonomy and that of a design methodology merge.

The purpose of a VHDL taxonomy is to provide a way of characterizing VHDL models in terms of a set of attributes and attribute values so that one can relate edges (information flow) in a workaholic to VHDL models that may either be produced by a design activity or obtained from a design reuse library. For this reason, the successful development of a VHDL taxonomy requires many things to be understood, for example:

a) Where does a particular type of model fit in the design process?

b) What design risks are reduced through a particular type of model's use?

c) What are the benefits /costs associated with the use of a particular type of model, for example, what errors will it detect, how much development and execution time are required, and so forth?

Many schemes have been developed to help us organize and think about the design process; most recognized amongst these is the Gajski-Kuhn Y-chart. In the VHDL community the Ecker-cube has obtained considerable attention. The VHDL taxonomy activity being undertaken by the RASSP program is building upon these earlier efforts. The attributes and attribute values that are used to characterize a VHDL model lie at the heart of this activity. Ideally, this set of attributes would possess two properties:

> **Property 1**: It would be desirable if two VHDL models that differ in at least one attribute value would be used by different activities in the design process or be used by one design activity for distinctly different reasons.
>
> **Property 2**: It would be desirable if a model could be automatically categorized by automatically determining its attribute-values.

The first property would support RASSP's drive to reduce development time by a factor of four by facilitating VHDL model reuse and model interoperability. Theoretically, the first property ensures that an isomorphism exists between the design methodology and the VHDL taxonomy. This fact ensures that a VHDL reuse library can be organized so that designers working on a particular activity can easily obtain VHDL models that are potentially relevant to what they are working on by requesting models with certain attribute values. For instance, a designer that is developing the architecture for an embedded system is interested in quickly and easily identifying where bottlenecks for a particular architecture exist and performing "what-if" analysis until a well-balanced system architecture that meets the cost and form-fit-function constraints of the system is identified. In this process, access to VHDL models that possess certain qualities that can be used to help evaluate evaluate the system’s performance is required. Today, such models are often referred to as "Performance Models."

So, you might ask, why bother with a taxonomy (determining a set of attributes and attribute values for characterizing different types of models). Why not, for instance, just call performance models, "performance models," and when a designer needs performance models he requests "Performance Models" from the reuse library. Unfortunately, the simple solution implied by this question may not be appropriate. Given the term "Performance Model," different individuals are likely to have different understandings of what a performance model is. As Fred Rose from Honeywell Technology Center writes, "As the EDA industry increasingly moves to a ‘plug and play’ business model, a common language or ‘lingua franca’ is essential. But more than a common language is required; common usage or style is also required." In essence, the intent of this common style is to ensure that models of a certain type possess certain characteristics or attributes. By developing a taxonomy that identifies the appropriate set of attributes for characterizing VHDL models and identifies the attribute values a particular model type should possess, the taxonomy provides a more rigorous definition for a model type than a simple name. This rigor also enhances the interoperability between two different models of the same type by ensuring that each model of a certain type exhibit a shared set of characteristics. Model interoperability and therefore “plug and play” can be further enhanced by adherence to Property 2.

By satisfying Property 2, the taxonomy can provide the basis for automating the process of determining whether a given model adheres to some modeling style or not. Because the development of modeling guidelines is the approach being taken within the VHDL community to address the model interoperability or "plug and play” issue, automatically determining whether a given model satisfies a style's requirements can provide significant benefit to the RASSP community.

However, defining a taxonomy for VHDL models that satisfies this second property is a significant undertaking because it requires the taxonomy to be formally specified, for it is only through a formal machine processable definition that Property 2 can be realized.

The RASSP Program's efforts to develop a taxonomy for VHDL models is one of the critical elements being addressed by RASSP that will enable effective top-down system level design with VHDL. The development of a VHDL model taxonomy will contribute to the widespread usability of system level VHDL models and provide the basis for sharing hardware design knowledge in the form of reusable VHDL libraries. A great deal of excellent work has already been done in this area by Dr. Harr of ARPA, Prof. Madisetti of Georgia Tech, Carl Hein of Martin Marietta, Paul (Kassey) Kalutkiewicz of Lockheed Sanders and a score of others too numerous to name. However, a great deal remains undone. As Winston Churchill once said, "...this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning." So it is with RASSP's efforts to define the conceptual structure for organizing VHDL models by defining a VHDL taxonomy.

contact RASSP E&F by the World Wide Web
at URL http://rassp.scra.org

Announcing Summer '95 RASSP Short Courses

The RASSP Education & Facilitation team will offer two short courses on key advances of the RASSP program. Each of these four-day courses is intended for design engineers and will provide hands-on exercises in addition to the lectures.

Rapid Prototyping Methodology Using VHDL

Topics Include:

-VHDL Basics
- Behavioral VHDL
- Structural VHDL
- System-level VHDL
- RASSP Methodology Overview
- System Level Modeling
- Hardware Synthesis
- Test Technology Overview
- Libraries
- Virtual Prototyping using VHDL

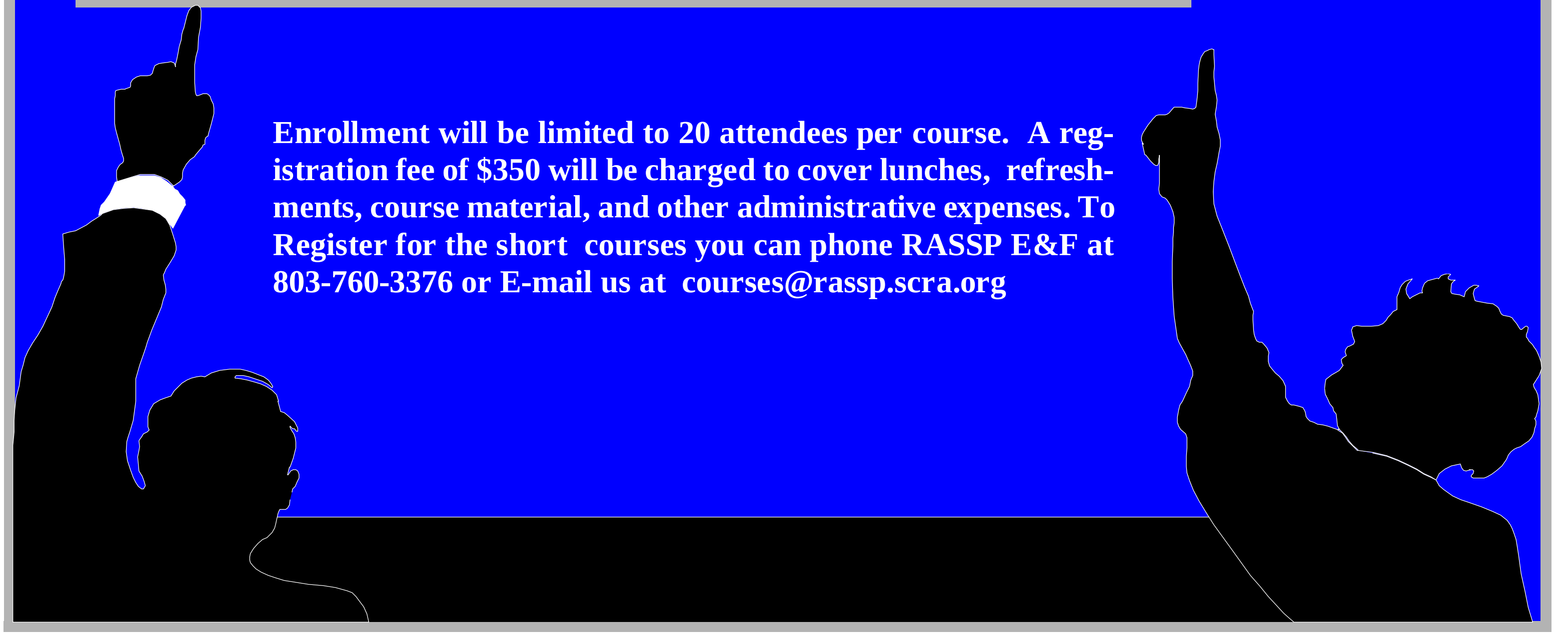
Led by: Prof. J. Aylor, University of Virginia
Course Dates: August 7-10, 1995
Location: Boston, MA

Algorithm and Architectural Design for Embedded DSP Systems

Topics Include:

- RASSP Methodology Overview
- DSP Architectures
- Algorithm/Functional Design for DSP
- HW/SW Partitioning
- Scheduling and Assignment for DSP
- Communication and I/O Architectures
- Real Time Systems
- Virtual Prototyping for DSP Architectures

Led by: Prof. V. Madisetti, Georgia Institute of Technology
Course Dates: August 22-25, 1995
Location: Tuscon, AZ



Enrollment will be limited to 20 attendees per course. A registration fee of $350 will be charged to cover lunches, refreshments, course material, and other administrative expenses. To Register for the short courses you can phone RASSP E&F at 803-760-3376 or E-mail us at courses@rassp.scra.org

RASSP Digest-Rapid Prototyping of Application-Specific Signal Processors

The RASSP Digest is published quarterly and provides information for and about the RASSP Program and rapid systems development. For more information, contact Dr. Anthony Gadient or Dr. Vijay Madisetti, Editors, at the addresses below:

Dr. Anthony J. Gadient
Phone : 803-760-4082
FAX: 803-760-3349
Email: gadient@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

Dr. Vijay K. Madisetti
Phone: 404-853-9830
FAX: 404-853-9171
Email: vkm@ee.gatech.edu
Georgia Tech
Sch. of Elec. & Computer Eng.
Atlanta, GA 30332-0250

Tommy C. Taylor
Managing Editor
Phone: 803-760-3792
Email: taylor@scra.org

RASSP Steering Committee

ARPA (ESTO)	
-Mark Richards	***Program Manager***
-Elliot Cohen	
ARMY (ARL/EPSD)	
-Clare Thornton	
-Randy Reitmeyer	***Administrative COTR, Martin Marietta***
-Arne Bard	***Technical COTR, Martin Marietta***
NAVY	
-Ingham Mack (ONR)	
-Gerry Borsuk (ONR)	
-Joe Killiany (NRL)	***Administrative COTR, Lockheed/Sanders***
-J. P. Letellier (NRL)	***Technical COTR, Lockheed/Sanders***
AIR FORCE	
-Bill Edwards	
-Stan Wagner	***Technology Base and Facilitation/***
-John Hines	***Educator COTRs***

Calendar of Events

VHDL International Users Forum *For More Information: VIUF* *415-329-0578*	*April 2-6, 1995*	*San Diego, CA*
32nd Design Automation Conference *For More Information: MP Associates* *303-530-4333*	*June 12-16, 1995*	*San Francisco, CA*
2nd Annual RASSP Conference *For More Information: Mark Feuchter* *703-351-8463*	*July 24-27, 1995*	*Arlington, VA*

SCRA
5300 International Blvd.
N. Charleston, SC 29418

RASSP Digest

RASSP - *Rapid Prototyping of Application Specific Signal Processors*

Vol.2, 2nd. Qtr. 1995

4X - Charting the Course

Methodology

RASSP
Reinventing
Electronic
Design

Architecture

Infrastructure

ARPA • Tri-Service

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • MMG • ADL

In This Issue

Editor's Corner

Prime Development

Benchmark Program

Technology Base

RASSP VHDL Working Group Update

In This Issue

by Anthony Gadient

This issue of the RASSP Digest focuses on the ambitious goal RASSP has set to provide a 4X improvement in time-to-market, life-cycle cost and design quality. The methods by which RASSP will achieve these improvements is presented together with the progress that RASSP has made towards achieving these ambitious objectives. Papers presenting an analysis of current practices and ways of measuring the improvements to current practice are provided. The RASSP Digest editors hope the readers will find this issue of the RASSP Digest both informative and beneficial.

Don't forget to mark your calendars to attend the 2nd Annual RASSP Conference, July 24-27, 1995 at the Hyatt Regency Crystal City in Arlington, Virginia. To receive more information or reserve your spot at the conference, contact Mark Feuchter, System Planning Corporation, 1429 N. Quincy Street, Arlington, VA 22207 or via Internet at the following addresses:

rassp_conference@arpa.mil
http://rassp.scra.org

An Approach to 4X

by Vijay Madisetti

This editorial highlights four main advantages of the RASSP approach that the author believes will result in significant improvements to current practice:

- **Top-Down Design Methodology Using VHDL and the Use of Virtual Prototyping at Various Levels:** The prototyping time identified by the RASSP E&F current practice (1993) model will be reduced by 25% using a top-down design methodology. In addition, the benefit of the top-down design methodology towards rapid legacy system upgrades is very promising. The new ideas of executable specifications and requirements will have consequences on the design methodology of systems of the future.

- **Reuse Design Libraries for Application Specific HW/SW Components:** The extensive reuse of past design experience (in an automated manner) is expected to lead to improvement in algorithm design, architecture selection, performance evaluation and effective HW/SW integration through error detection and correction. The expected improvement due to reuse is a 20-25% reduction in the development time identified by the RASSP E&F current practice (1993) model.

- **Automation and Enterprise Integration:** These capabilities are rapidly improving as commercial CAD vendors are attempting to provide interoperable tool suites and high-level and behavioral synthesis environments. These effects are aimed at rapidly and effectively raising the level of design abstraction leading to impressive gains in productivity. The capability for paperless design specification and reuse, together with concurrence control and effective teamwork capabilities through improved automation and design environment integration is expected to improve prototyping time of the E&F current practice model by 20-25%.

- **Customer Satisfaction and Virtual Corporations:** The use of virtual prototyping allows rapid customer evaluation of designed prototypes without incurring expensive hardware (or FPGA-based prototyping) costs. All design and implementation is done entirely in software prior to EMD. The flexibility of customer input leads to rapid incorporation of design specification changes or upgrades into most stages of the design process. For example, a change of word length from 16 to 18 bits can be incorporated into a large design even at the final design stages (HW/SW integration) in a matter of a few hours by changing the VHDL architecture files followed by recompilation. The new design process ensures that multi-corporation teams can be rapidly put together by extracting specialities of various teams to harness them on one application rapidly and efficiently through the use of cooperative teamwork-based design environments and the sharing of design experience through portable reuse libraries.

Thus, RASSP is providing a low risk process flow for large system design, especially in fixed-price contracts where the vendor runs significant risks of cost overruns for underestimating the design complexity management. Another possible advantage to using the RASSP approach is the elimination of the DEMVAL stage of current practice model due to increased confidence in the virtual prototyping process. These gains, though difficult to quantify, will be of enormous significance to the operation of virtual corporations of the near future.

Acknowledgments

The editor acknowledges with gratitude inputs from various RASSP participants and programs. However, the views expressed in this editorial are the author's own, and do not necessarily represent the viewpoints of the US Department of Defense, ARPA, MIT-Lincoln Laboratories, Lockheed Martin-ATL, Lockheed Martin-Sanders, Raytheon, Hughes, Motorola, or SCRA.

Advanced Technology Laboratories' Path to 4X Improvements

by Jeff Pridmore

To provide 4X improvements in time-to-market, life-cycle cost, and design quality, Lockheed Martin's Advanced Technology Laboratories is uniquely combining the three elements of the RASSP technology triad -- methodology, Model Year Architecture, and design environment -- into an integrated, rapid prototyping environment.

1. Problem

Today's military designs are the product of a long, serial design cycle that has limited ability to respond to changing requirements and technology. Current practice surveys -- as detailed by Madisetti and Corley in this issue of the RASSP Digest -- indicate that today's developments require anywhere from 37 to 73 months. Commercial processor technology offers significant capability upgrades every two to three years, yet typical military development/deployment cycles are more than five years. The result technology in the fielded system is one to two generations behind the state-of-the-art at the time of deployment.

Technology obsolescence is further exacerbated over the life cycle of the system. The platform life cycle for military systems is often more than twenty-five years, and the DoD focus is on further extending life cycles. Systems can thus be upgraded 8-10 times over their operational lifetime, with about half of these required to meet new operational requirements.

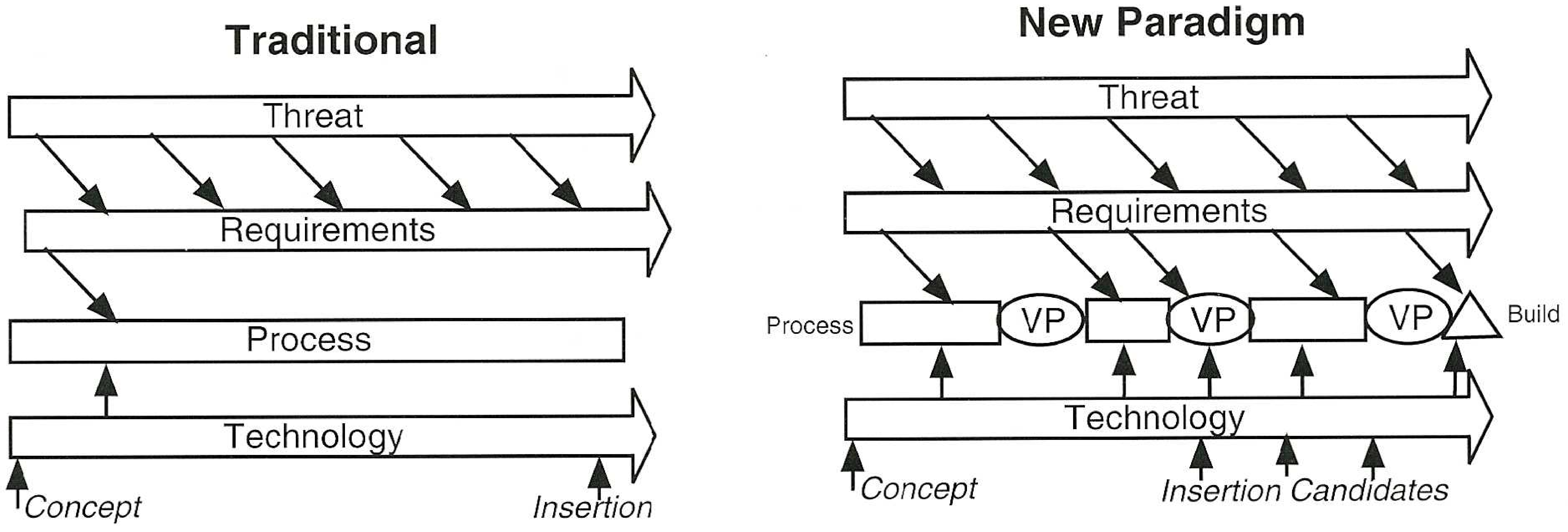
2. The RASSP Approach

RASSP's Model-Year concept strives to fundamentally change the design process from a custom-oriented, serial design approach, shown in in the left side of Figure 1, to the iterative, simulation-based approach, shown on the right side of Figure 1. The result is a series of virtual prototypes (fully simulated design implementations) that can be built quickly for insertion into products as new developments or upgrades.

The virtual prototype is developed as an evolving executable specification, which is a form ofan information model composed of these:

- Functional and performance simulation models,
- Testbenches,
- Requirements -- size, weight, power, cost, etc.,
- Process description -- previous and upcoming steps in the design process along with engineering notes, lessons learned, etc..

Lockheed Martin evaluated existing practices and defined the changes required to implement a 4X improvement. Figure 2 shows the schedule for an actual 48-month radar upgrade program, with the time required for concept development, architecture trade-offs, detailed design, manufacturing, and integration and test. Note that the schedule also includes a 10-month redesign cycle for unanticipated reworks that occur due to either design errors or functional updates uncovered during integration and test. The second set of schedule elements shows the improvements RASSP must provide to accomplish 4X. Evaluation of the figure indicates several key points. Under RASSP, a higher percentage (about 1/3 of the 14.5 months) of the overall development cycle is spent during concept design and trade-offs than in current practice (about 1/4 of the 58 months).



Traditional Designs
-- Static, sequential process (waterfall model)
-- Custom designs
-- Technology dated when fielded
-- High design (NRE) and life cycle costs (LCC)

RASSP Virtual Prototyping (VP)
-- Dynamic, risk-driven concurrent process (spiral model)
-- Incorporates evolving requirements
-- Rapid insertion of COTS technology
-- Low-cost insertion, LCC

Figure 1. Current design approach versus the RASSP Model-Year concept

Where will the payoff come from?

- **Big Hitters**
  - -- Improved quality
    - - Eliminates redesign/ fabrication
    - - Drastically reduces integration/ test
  - --Design reuse
    - - Reduces manufacturing/ detailed design
- **Key Contributors**
  - -- Productivity is the key contributor
    - - Shrink system and architecture trade-offs
    - - Concurrent access to information/tools improves productivity across the board

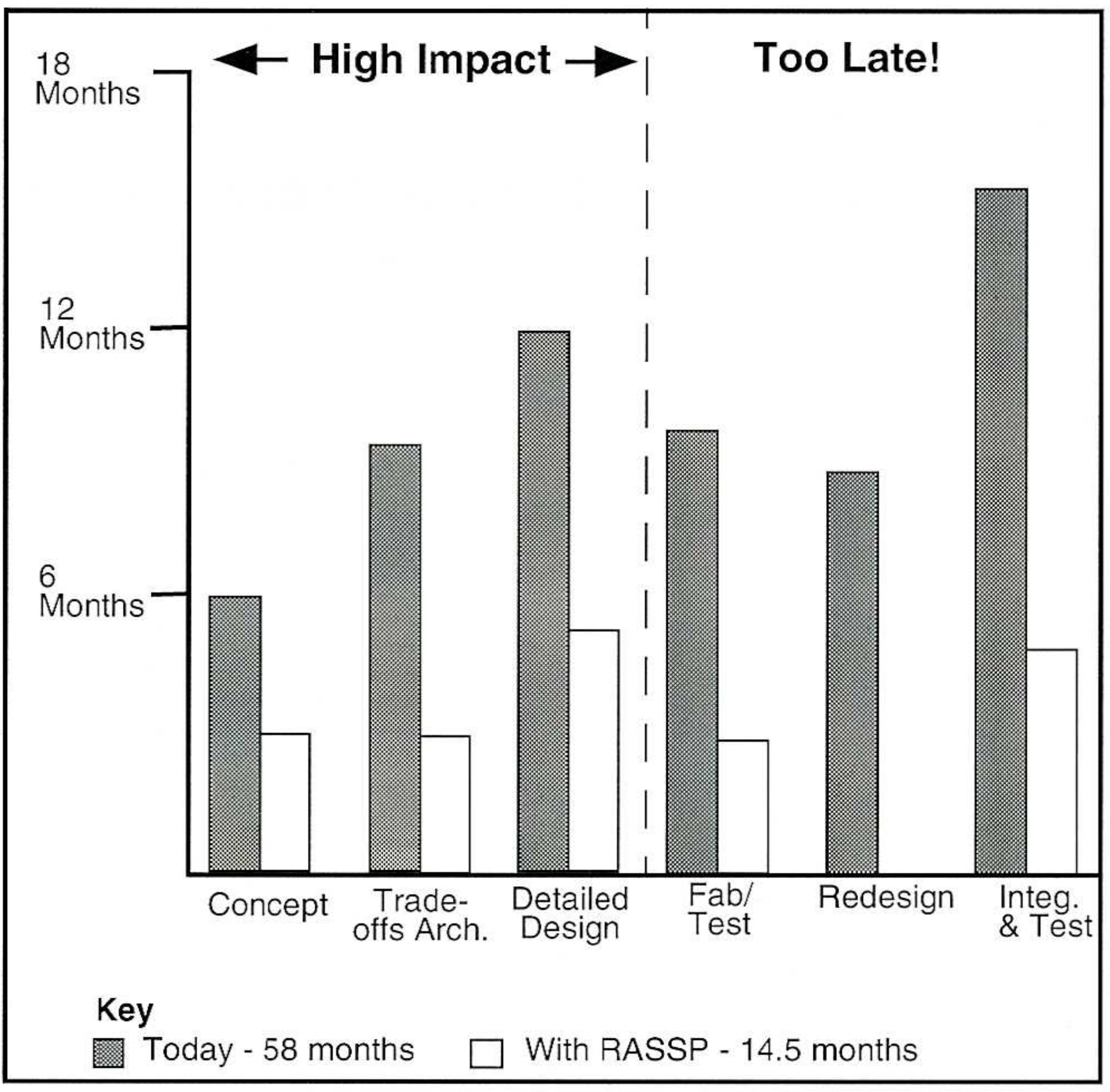


Figure 2. Current practice versus RASSP process comparison.

Less time (1/4 of the 14.5 months) is spent in redesign, integration and test for RASSP than for pre-RASSP approaches (41% of the 58 months).

In all cases, RASSP promises to provide signifcant cycle time improvements -- but what are the elements that lead to this payoff? First, the only way to achieve 4X is to eliminate redesign and drastically reduce the integration and test time for systems. Simply improving design efficiency will not achieve 4X; the design portion of the process only accounts for 50 percent of the total task. Second, detailed design and manufacturing times must be reduced to further lower cost and shrink schedules.

Providing the gains described in Figure 2 requires these actions:

- Implementing concurrent design practices using integrated hierarchical design verification to improve design quality and performance,
- Maximizing reuse of both hardware and software elements to dramatically decrease cycle time,
- Improving productivity enables rapid implementation of the steps that lead to improved quality, such as hardware/software codesign and virtual prototyping.

Lockheed Martin is quantifying the time-to-market and life-cycle cost impact of these elements by developing a parametric model of the RASSP process using the PRICE cost estimation tool. We have modeled a typical airborne radar example for both current practice and the projected RASSP process. While this work is still ongoing, sensitivity analyses using the tool emphasized the above conclusions. Our simulations show that reuse will probably account for more than half the projected improvements. The second major factor is productivity, which enables many of the quality elements described above, and which can also supply up to a factor of 2X. We are still introducing new RASSP concepts into the price model, such as the effects of virtual prototyping (enabling first-pass success) on schedule and cost, and the impact of the Model-Year upgrade approach on long-term life-cycle support costs. Our studies in these areas will further quantify the impacts of the technology developments and provide a more robust model of the RASSP process. A robust model of this type is very important to help potential users project the improvements in time-to-market and life-cycle cost they can expect out of RASSP -- not all projects will be able to realize 4X. Factors that affect improvements include availability of models to support virtual prototyping, the amount of reuse applicable to the project, and domain-specific elements that dictate specific approaches.

3. RASSP Technology Enablers

The three major impact areas -- quality, reuse, and productivity -- are closely linked to the RASSP technology triad, as shown in Table 1. All of Lockheed Martin's ongoing technology developments that support these impact areas cannot be describedin this article; several key developments highlighted in the table are summarized in the following paragraphs for each area of the technology triad.

We expect several key elements of the RASSP Methodology that depart from current practice to be large contributors to the 4X improvements:

- Processes to support design efficiency and reuse, including application of object-oriented analysis and design techniques, and maximum use of reuse libraries,
- Maximizing use of concurrent engineering, including the concept of Integrated Product Development Teams; we emphasize methods to enhance both concurrency and collaboration between tasks throughout the design cycle,

- Applying the spiral development model to signal processor design, providing a risk-driven, iterative approach to rapidly developing prototypes,
- Creating a new architecture process that implements hardware/ software codesign to hierarchically design and verify the signal processor hardware and software design throughout the design cycle. The result is a fully operational virtual prototype before manufacturing release.

In conjunction with Methodology, the **RASSP Model Year Architecture** is a framework for hardware and software reuse. It provides the design guidelines and constraints for implementing signal processors to support efficient upgrades using modular building blocks. Lessons learned from early programs in software reuse showed that placing elements in a library does not create an environment that supports reuse; the approach must be an integrated, fundamental part of the overall design process. The Lockheed Martin Model Year Architecture provides resources and constraints to the design process and enforces a structured approach that implements scalable, modular hardware and software processing elements in a functional architecture. An instantiation of the functional architecture results in a set of encapsulated library elements. Encapsulation refers to structure added to otherwise "raw" library elements to support the functional architecture framework and ensure library element interoperability and upgradability.

Table 1: RASSP technology triad developments supporting 4X improvements

	Resue	Improved Quality	Efficiency
Model Year Architecture	Virtual Interface stds Re-use framework Application notes / development guidelines	Validated HW / SW libraries	VHDL / HOL reuse templates / toolkits
Methodology	Integration of MYA into process	Risk_driven interative design	Steamlined processes
	O-O design approach	HW / SW codesign	Concurrent / Paralleized tasks
Infrastructure -EnterpriseSystem	Library management system	Workflow magagement	Distributed interaction Integrated data management
- Design Environment	Graph-based SW generation	Virtual Prototyping	HW / SW codesign tools

The Model Year Architecture also provides a set of design guidelines and constraints for general architectural development, such as how to properly use the functional architecture framework, general use of encapsulated libraries, and, most importantly, procedures to encapsulate new library components. These design guidelines and constraints are incorporated into the RASSP design methodology.

The RASSP approach to implementing the Model Year Architecture is based on modular, scalable architectures that use functional standard interfaces. By standardizing on functional interfaces, we can maximize independence from technology (electrical versus optical) and specific hardware versus software (processor-based versus dedicated hardware). We are using this approach to define several **Standard Virtual Interfaces** that define functional VHDL encapsulation wrappers, which enable modules to be seamlessly interconnected with minimal performance impact.

The RASSP **Enterprise System** provides the overall infrastructure to support Integrated Product Development Team interaction. The Design Environment provides the tools to implement end-to-end virtual prototyping. The enterprise system is the framework for effective integration of the software tools and models used to develop and manufacture RASSP products. The elements of the enterprise system that most contribute to 4X improvements are the Design Methodology Manager (DMM) and Product Data Management capabilities.

The DMM guides users through the process, providing access to the appropriate tools at the proper times, ensuring that complete data packages are generated and that all critical steps in the process are followed. This capability will greatly improve the quality of designs to ensure first-pass design success. Seamless tool access throughout the design process is also provided. DMM triggers the appropriate elements within the Product Data Management System to ensure that data automatically transitions between workflow steps in the proper format. By abstracting engineers from the details of the tool interaction and data management, large productivity gains can be realized and error-prone manual data translations eliminated.

The Design Environment strives to provide a seamless set of end-to-end tools to support hardware/software codesign and full design verification before manufacture via virtual prototypes. The tools being extended on the RASSP program that we expect to have the largest impact are those that support hardware/software codesign, and these include architecture trade-off, architecture-level design verification, and automated software generation tools.

The **architecture selection tools** enable users to select and size a processor architecture based on processor requirements generated during the subsystem design phase. Users first partition the functionality between hardware and software, and then verify top-level functionality using algorithm-level tools, such as Matlab, SPW or PGSE and evaluate the overall timeline performance using tools such as JRS's Netsyn. Other important factors, such as size, weight, power, cost, reliability, etc., are also included in the trade-offs at this level through a tool suite integrated into an architecture design advisor.

Once users have selected a candidate architecture, it is verified using the **architecture verification** tools, as shown in Figure 3. This tool suite consists of a set of performance and functional simulators at various levels of design abstraction (abstract behavior, ISA-level, Register Transfer Level, etc.) and models of computation (data flow, control flow, event driven, etc.) that are iteratively invoked by users to hierarchically verify the processor design before detailed implementation. Emulation and hardware testbed capabilities will also be integrated into the capability by combining simulation backplane and mixed-level domain

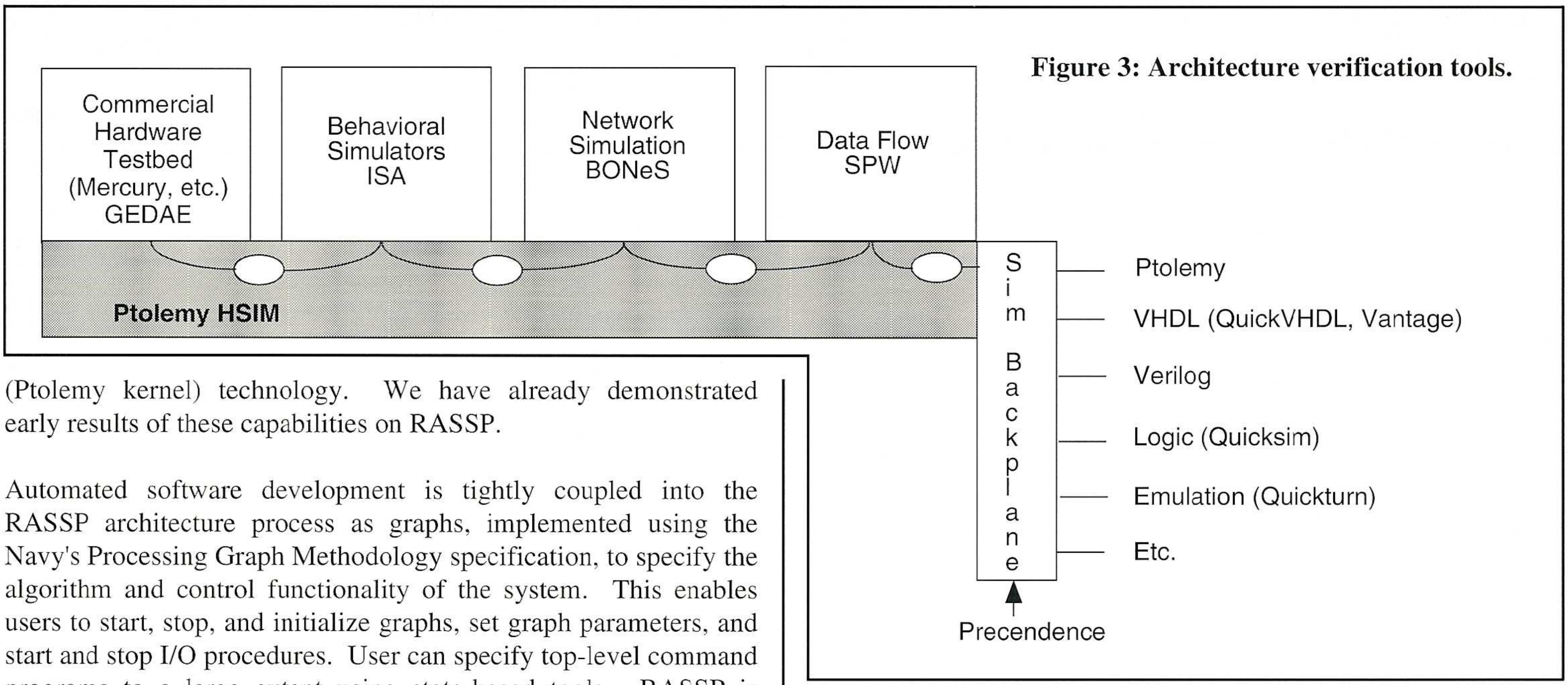


Figure 3: Architecture verification tools.

(Ptolemy kernel) technology. We have already demonstrated early results of these capabilities on RASSP.

Automated software development is tightly coupled into the RASSP architecture process as graphs, implemented using the Navy's Processing Graph Methodology specification, to specify the algorithm and control functionality of the system. This enables users to start, stop, and initialize graphs, set graph parameters, and start and stop I/O procedures. User can specify top-level command programs to a large extent using state-based tools. RASSP is implementing a set of autocode generation tools that will enable users to take PGM graphs and automatically generate downloadable code for embedded multiprocessor environments. These tools implement the Model Year Architecture by using the reusable software libraries and targeting the code generation to support the Model-Year application programming interface (API) and run-time system (RTS).

4. Conclusions

The Lockheed Martin RASSP team is providing fundamental technology improvements via the RASSP technology triad to enable 4X improvements in design-cycle time, quality, and life-cycle cost reduction. We are demonstrating initial implementation of our reuse-based methodology implemented in an enterprise-wide system that supports distributed, collaborative interaction. We expect the Model-Year paradigm to be able to demonstrate 4X improvements over today's custom-based design approaches well before the end of the program. While the performance improvements across a wide range of programs will vary based upon the application, type of development, and availability of models, large improvements are already being demonstrated for ongoing programs.

Road to 4X

by Larry Scanlan and Leroy Fisher

1. Introduction

The RASSP Program has ambitious goals: 4X decrease in product development cycle-time, 4X decrease in life-cycle costs and 4X increase in product quality. Reaching these goals requires a map, so we can choose the route that leads to the desired destination while avoiding financial mountains too high to climb or technology gaps too wide to jump. With the map we can expend all of our energy navigating the routes that will accelerate our progress.

In the sections that follow, the factors that contribute to cycle-time and quality will be identified along with the barriers to change. Next we describe some of the approaches being taken by the Lockheed Sanders RASSP team to address each of the factors and associated barriers. We will then summarize the results of a product development task analysis to show how each RASSP process reduces cycle-time and improves quality. Finally, we will assess the progress of the IRST Signal Processor Demonstration Team towards achieving 4X.

2. Contributors to Cycle-time, Cost and Quality

The fishbone chart shown in Figure 1 identifies the major factors contributing to cycle-time, cost and quality. Each will be described briefly in the following paragraphs.

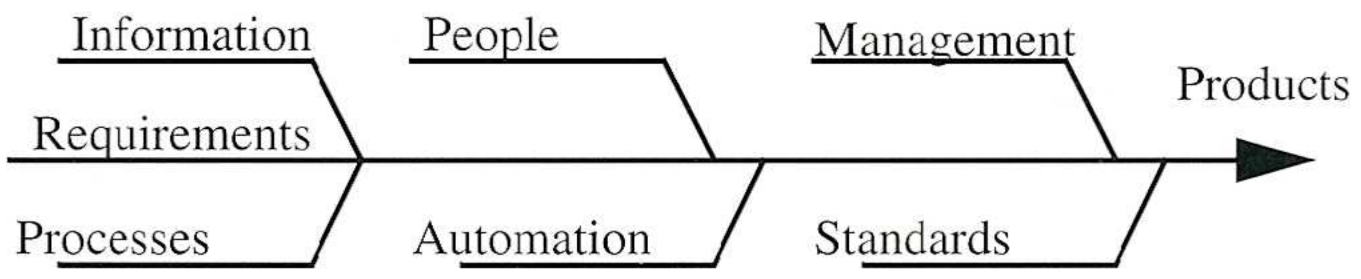


Figure 1. Fishbone chart showing the significant contributing factors to cycle-time, cost and quality

.

2.1 People

People represent a critical element in the quest for 4X. People make decisions, people apply expertise, people enable, people obstruct, people create, people destroy, and people drive the process. The inherent adaptability of people makes them indispensable to creative activities. At the same time, the capacity for independent function and innovative approaches result in high levels of variability whenever people contribute to a process. To harness the creativity and minimize the variability, people need ready access to all the necessary facts and data, must be trained and must be empowered to make decisions and act on those decisions in an open and disciplined way.

2.2 Processes

Processes -- both business and engineering -- form an essential foundation to the establishment of predictable cycle-time, cost and quality. They must be defined, understood and institutionalized to be effective. The consistent application of process across the organization and across time reduces variability and, with suitable measurement, provides a systematic method for continuous improvement.

According to the Boston Consulting Group [1], "structural sources of competitive advantage such as ... low cost production ... or technology are no longer enough. Companies win by having business processes that recognize and meet customer needs fastest." Hammer and Champy in their book "Reengineering the Corporation" [2] cite several case studies where business process improvements and information technology have been combined to achieve results far beyond 4X. Clearly, process is an essential enabler in the quest for 4X.

2.3 Automation

Automation frees people to use their energy and creativity to solve problems and to continuously improve. However, as the saying goes, automating a bad process will only give you bad results more quickly. The science of knowing when and how to automate will be as important as the technology of automation.

Automation includes applications that accomplish single tasks such as logic simulation as well as infrastructure tools that enable communications, information management, process management and so on. It is sometimes useful to make the distinction between domain-specific applications and cross-domain applications where domains can represent specific engineering disciplines or organizational entities.

2.4 Information

Information and its reuse are vital for faster and better product development. Knowledge, the essence of information, must be easily preserved as it is created and even more easily made available when needed (reuse). Information includes rationale, metrics, product information, component information, process information, resource information, and market information to enumerate only a few of the many information categories.

The reuse of information keeps us from reinventing every time a new design problem comes up. To reuse something you have to have first captured the data. The more the collection of this data can be made either automatic or an easy part of working, the more complete will be the database for future reuse. Once captured, the information needs to be made available to the engineer or manager in a manner that makes reuse easier than starting over. When finding the information is perceived as more difficult than starting from scratch, the carefully captured information will have no value. The database and user interface must be robust and the search engine powerful to take advantage of reuse. The RASSP Design Environment can do a great deal in this area.

2.5 Management

Management represents the key decision-making element that can either make things happen or bog things down. Situational awareness, the ability to provide the right information at the right time, is a key enabler of the management function. Management is also a central force in institutionalizing processes and ensuring their consistent application.

Management is responsible for constructing plans and forecasts, for acquiring and allocating resources, for ensuring the growth of the people, and for making sure the business remains profitable. This authority means that management bears a large portion of the responsibility for the success or failure of the organization in reaching 4X.

2.6 Standards

Standards, while difficult to establish, have the capability for significantly accelerating product development. Standards for the representation of data make information sharing, reuse, and long term support across an entire industry easy.

3. Barriers to 4X

Change is difficult, and a number of barriers need to be overcome to accomplish our 4X objectives.

3.1 Cultural

Existing company and institutional cultures represent one of the most significant barriers to improvement. Cultural barriers exist in the form of resistance to change, overly localized perspectives, and reluctance to be measured.

3.2 Financial

Financial barriers hamper the ability of an organization to acquire new technology, to field new processes, and to train and motivate the workforce. Failure to plan for and invest in new design automation technologies, such as high speed simulators or VHDL-based design techniques, can result in longer development cycle-times, lower product quality and higher product costs.

3.3 Technological

Technology barriers can impede progress on both the product roadmap and on the process or operations roadmap. On the product side, these barriers inhibit or delay the introduction of new products. On the process side, desirable changes in the way activities are performed will be delayed because the supporting technology is not mature. For example, synthesis from Behavioral VHDL has not been widely available in the past and has limited the process options for top-down design.

3.4 Informational

Even if the culture supports change, the resources are there to finance it, and the technology is available on the shelf, information must be present to trigger necessary change. Situational awareness is a key element in supporting managed change in both process and product.

Lack of information, wrong information or poorly timed information can all contribute to inappropriate tactical and strategic decisions. Part of the information barrier is inadequate experience, poor or improper training and lack of awareness that help might be available. This results in re-inventing and re-learning rather than re-using.

4. Key Elements Leading to 4X Improvement

The Lockheed, Motorola, Hughes and ISX RASSP Team is developing a balanced set of approaches to address each of the factors that contribute to improvement and the barriers that impede improvement. This balanced approach involves developing new engineering and business processes and new technology as well as improving access to resources and information.

4.1 Top-Down VHDL Design

Top-down VHDL design forms the cornerstone of the product design process. It begins with development of VHDL models for entire systems and continues onto hardware/software partitioning and through detailed hardware design and development. These models comprise the Virtual Prototype(s) of the system and system elements.

Top-down VHDL design is most effective at compressing the timeline within distinct phases of design. Additionally, using VHDL as the carrier of product information and product intent significantly lowers the information barriers between design phases and provides a firm basis for supportability as a product is fielded. This significantly reduces the rate at which errors are introduced into the design, enables early integration of hardware and software and leads to reduced lifecycle costs.

Top-down VHDL design leads to higher product quality through the unbroken thread of product functionality from final design back to original requirements. In addition, early use of high-level VHDL models allows a larger design space to be explored and evaluated, enabling more informed trades between cost, time and function.

4.2 Structured Software Development

Structured software development complements the VHDL-based hardware design process, supporting hardware software codesign. The two together enable modular product design thereby facilitating design for reuse; the precursor to reuse of design.

Structured Software Development, like Top-Down VHDL Design, is most effective at compressing the timeline within distinct phases of design. And, like VHDL, the use of standard languages (Ada, for example) significantly lowers the information barriers between different phases.

4.3 Integrated Product Development (IPD) and Virtual Corporation Technology

Integrated Product Development teams have all of the disciplines needed to accomplish product development from concept to field support working as a single integrated team to efficiently and concurrently create new innovative products. The team approach enables tight linkages between hardware, software, product design, manufacturing, procurement, reliability, maintainability and supportability to be established and maintained.

IPD can be made significantly more powerful with the addition of tools and processes to enhance situational awareness. The RASSP Design Environment Prototype (RDEP), first shown at our six-month review and updated regularly since that time, incorporates several situational awareness tool concepts. Workflow management, process management, automatic notification and a common desktop environment are some of the tools that offer potential for making sure everyone has access to the facts necessary to make informed decisions. We are also developing methods to extend our situational awareness capabilities further and have plans for experimenting with several alternatives to assess their effectiveness.

Virtual corporation technology extends the concept of IPD to encompass multiple companies, geographically separated to perform as if they were a single company located in a single location. Virtual corporation technology allows the flexible creation of teams comprised of electronically co-located workers and addresses both engineering and management issues. It includes these:

- Coordination technology
- Electronic information exchange
- Cross company secure access to design automation
- Cross company secure access to expertise
- Cross company secure access to reuse information

4.4 Reuse Databases and Libraries

Our RASSP Team recognizes the importance of reuse and reuse libraries. We are capturing the VHDL elements from the IRST Demonstration Model Year 0 in a database and will be demonstrating their reuse in Model Year 1. Our experiences will be valuable in assessing the additional requirements for easy reuse. We are also demonstrating how processes can be reused by applying parts of the Model Year 0 process to Model Year 1. Because we learned a great deal during Model Year 0 and because Model Year 1 has added complexities such as legacy system considerations, the process has been refined and tailored to meet the needs of the next IRST Demo. Our experiences in trying to reuse process elements will further our insight into process reuse.

As with situational awareness, the RDEP has been used to capture our reuse concepts and to act as an interactive requirements tool. We are currently conducting user evaluations of the RDEP, including feedback on our plans for reuse databases and reuse libraries. This feedback combined with our actual experiences using the RDE during Model Year 1 will verify our approach and suggest ways to improve.

4.5 Rapid and Disciplined Process

The RASSP Rapid and Disciplined process features mechanisms that support overlapping, concurrent activities where information flow between activities is tagged with an estimate of its maturity. This methodology makes the sharing of in-process information easy but more disciplined than in the basic IPD structure. This enables down-stream activities to begin work, as appropriate, with preliminary information. The RASSP document management and product data management facilities have been designed to support a disciplined process of review and promotion.

4.6 New Technology Development and Adoption

The RASSP team is approaching new technology in several ways:

- On-going evaluation of key technology areas
- Coordination with the RASSP-funded Technology Base
- Joint efforts with the EDA supplier community
- Development of management technologies that facilitate adoption of product technologies

Even without RASSP, technology growth will result in significant productivity improvements over the few years that RASSP is funded. We, however, are leveraging these "natural" advances to achieve dramatically better results much earlier.

5. Product Development Task Analysis

At the RASSP 18-Month Review held in El Segundo in February 1995, we presented a model for reaching 4X. This model was based on the results of a task analysis of the design process from very early system concept development and feasibility through development, test, production, and field support. Approximately 70 specific tasks were identified, and the associated duration, based on current practice, for each task was determined. The process flow was based on ADQAS (Advanced Design for Quality Avionics) [3] to ensure a disciplined methodology with a high probability of yielding exceptional product quality.

The baseline development timeline resulting from the assignment of duration to tasks was compared with the current practice model proposed by Vijay Madisetti and Jack Corley of the RASSP Education and Facilitation team and found to fit easily within their minimum and maximum timelines as can be seen in Figure 2. To make an equivalent comparison, it was necessary to correctly align the starts of the two timelines. Our timeline began two phases earlier than the E&F current practice model. In addition, the E&F current practice model stopped with E&MD while ours included production and out-year supportability upgrade. This favorable comparison helped increase our confidence that we had accurately captured the basic product development process. The only anomaly occurs in the time assigned to Preliminary Design. The E&F current practice model assigns less time to this task than we do, probably because of differences in our definitions of when one phase ends and the next phase begins.

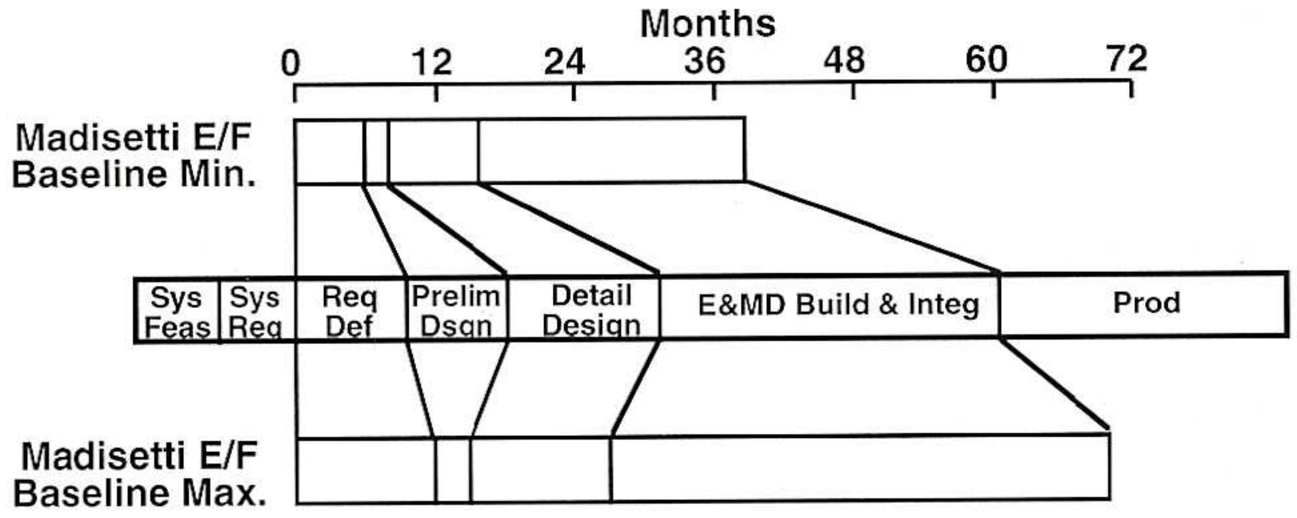


Figure 2. Comparison of Madisetti Current Practice Model and the Task Analysis Timeline.

We next applied the RASSP improvement elements discussed above to each of the 70 tasks to estimate how much reduction in task duration should result from the application of RASSP technology. In each case, an identification of which RASSP improvement elements were being applied and a rationale for why they should yield a reduction in cycle-time were codified.

Analysis of the task analysis data revealed three important conclusions:

- Achieving 4X requires more than within task cycle-time reduction.
- The early phases of the product design process are shortened the least while the later phases show the greatest benefit.
- Three elements contribute to more than half of the improvement.

The data show that a three times improvement in cycle-time can be expected by applying RASSP improvements to individual tasks. Achievement of the full four times improvement requires integration of individual tasks using the RASSP Rapid and Disciplined process to achieve effective task concurrency. Figure 3 graphically shows the 3X and 4X reductions in cycle-time.

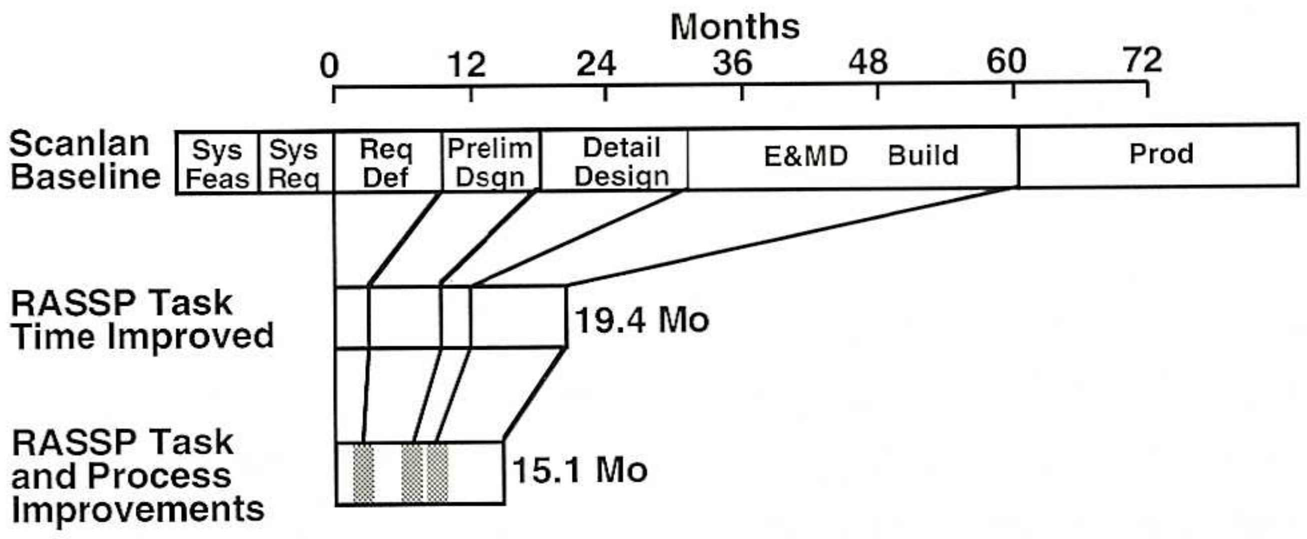


Figure 3. Within task improvements yield a 3 times improvement in cycle-time and the Rapid and disciplined Process applied across tasks provides the balance of the improvement to reach 4X.

An examination of Figure 3, reveals that some phases of the product development process are accelerated a great deal while others remain nearly the same in duration. In particular the Preliminary Design Phase takes nearly as long with RASSP as without. This is because the use of RASSP Top Down Design methods and Virtual Prototyping demand more work prior to PDR than the traditional methodology. However, the Detail Design Phase is substantially reduced because the Virtual Prototype has matured the design significantly. Similarly, the discipline and simulate-before-build-philosophy of RASSP make the E&MD Phase much shorter. This differs from a more traditional approach that allocates very large blocks of time to system integration and the correction of errors carried from the beginning of the design process.

Finally, we examined the data to determine which particular RASSP improvement elements were being cited most often and what proportion of the cycle-time reduction they contributed. Table 1 summarizes the top four improvement elements or factors. The largest contributor was Top Down Design using VHDL, which also includes Structured Software Development for programmable processing elements. This was expected and is consistent with the RASSP philosophy. Similarly, reuse was, as expected, an important contributor. Finally, team and management situation awareness was cited nearly as often as reuse and significantly more often than improved design automation tools. These data reinforce the observation that RASSP is not a tools program.

Table 1: Percent Contribution to Reduced Cycle-time by the largest four Improvement Elements.

Improvement Element or Factor	%
Top Down Design using VHDL & Virtual Prototypes	26
Reuse of Designs, Tools, Databases & Processes	18
Team / Management Situational Awareness	16
New / Improved Design Automation Tools	6
All Other Combined	34

6. Progress toward 4X - IRST Image Signal Processor

The Model Year 0 IRST Image Signal Processor development, undertaken as part of the Demonstration portion of the program, provides the first measure of how we are progressing toward the 4X goal. A comparison with the achieved schedule and cost of the IRST Demonstration with a similar program bid by Hughes in 1993 reveals a 2.2X improvement in both measures. The achieved design quality, measured as first time integration success, was not as good as it was expected to be. Everything that was simulated using the Virtual Prototype worked the first time. However, integration time was impacted by the need to correct errors in portions of the design that had not been simulated. Integration time was, however, less than that typically associated with a design of this complexity. While quality, measured as fitness for use, was high when the hardware was delivered to the Aircraft, there is clearly room for the additional improvements in quality that will lead to near zero integration time.

7. Conclusions

The work described above provides a roadmap for the RASSP Program to follow as we continue the development of the Process and Infrastructure that will yield a four-times reduction in cycle-time and cost with an equal increase in quality.

References

[1] "Reengineering and Beyond," Boston Consulting Group, 1993.

[2] M. Hammer, J. Champy, "Reengineering the Corporation: a Manifesto for Business Revolution," Harper Business, 1993.

[3] Hughes Aircraft Company Radar Systems, McDonnell Douglas Aerospace, "Advanced Design for Quality Avionic Systems," March, 1993.

RASSP Benchmark Program: Measuring 4x

by G. A. Shaw

1. Introduction

The primary goals established for the benchmark component of the RASSP program are

1. Evaluate performance of the RASSP design methodology relative to standard practice with emphasis on design cycle time, cost, and quality of products.
2. Identify weaknesses in the design methodology and supporting tools, and recommend corrective actions or improvements wherever possible.

Small design problems, nominally 6 months in duration and 5-10K hours of effort, are utilized as the primary vehicle for observing and assessing the performance of the RASSP process. The first two design problems, or *benchmarks,* are related to the design of a processor for synthetic aperture radar formation on board an unmanned air vehicle [1].

The 4X improvements sought through RASSP are to be measured relative to DoD contractors' standard practice at the start of the RASSP program (July, 1993). Therefore, a standard practice baseline must be developed for each benchmark as the basis for evaluating RASSP progress toward the principal goals of reduced design cycle time, reduced cost, and improved quality.

2. Some 4x Measurement Challenges

2.1 Limitations of Statistical Characterization

The principal metrics targeted for improvement by RASSP, time and cost required to produce an embedded signal processor, and the quality of the resulting product, are dependent on a multitude of variables. As evidence of this fact, note that commercial parametric cost estimation tools may require the specification of hundreds of parameters in order to estimate the cost and design cycle time associated with a particular hardware and software development effort.

The success of commercial parametric cost estimation tools provides evidence that, given a stable design process and many trials over which to make measurements, it is possible to develop reliable predictors of future performance. However, the RASSP process is continually evolving, and the users of RASSP are continually learning how to efficiently use the process, so that the development of a statistically significant database for calibrating performance of the overall process is not currently feasible. As a consequence, the most significant challenge for the benchmark activity is to develop *quantitative* metrics for assessing the performance of the evolving RASSP process over the half-dozen or fewer available benchmarks (trials).

The general approach adopted for the benchmarking is to compare performance of the RASSP process to parametric cost estimates for the same design problem (i.e., benchmark). The parametric cost estimates represent a statistically derived standard practice baseline. The result is a comparison of a single RASSP "trial" to an average of many standard practice "trials."

2.2 Interdependency of RASSP Objectives

A second challenge stems from the fact that some of the goals, such as reduced cycle time and development cost, might be achieved at the expense of other goals, such as improved product quality. For example, developing a design for thirty-year supportability with built-in test (i.e., improved quality and life cycle cost) will add to the development effort (design cycle time and cost) of a processor with otherwise equivalent functionality. Similarly, provision for model year upgradability and reuse of the application software (reduced life cycle cost) will add to the development time and cost of the first version of a processor.

As a consequence, a simple measure of development time or cost for a processor is not sufficient to demonstrate that RASSP is achieving the desired performance goals because it ignores too many other dimensions of the development process and requirements. Instead, one must compare the time and cost achieved using RASSP to the actual or estimated time and cost achievable using standard practice, assuming the same set of processor requirements and objectives, and the same level of expertise on the part of the Development teams.

2.3 Dependence on Experience of Personnel

The expertise of the benchmark execution team, and in particular the prior experience with similar processor designs, has a significant impact on the cost, schedule, and quality of a processor. The importance of the development team experience on a project is illustrated by applying the PRICE-H parametric cost estimation tool to estimate the development cost of a 6U VME form-factor board for SAR processing. Changing the parameter which defines the amount of prior experience in developing similar boards from "significant" to "none" approximately doubles the cost of the board. This implies that without resorting to any process improvements, it is possible to achieve a 2x reduction in cost merely through the choice of individuals assigned to the development.

Clearly these types of dependencies need to be accounted for by developing a baseline standard practice model which reflects the level of expertise and prior experience represented in the RASSP benchmark execution team.

2.4 Maturity of Technology

Contemporary embedded digital processor designs are almost exclusively comprised of integrated circuits, and it is the encapsulation of complexity within the integrated circuit that enables wide availability and application of sophisticated technology such as programmable DSP chips. However, in the 1960s, if one were to measure the cost-effectiveness of developing a digital design exclusively with integrated circuits, the results would suggest that the technology was not cost effective simply because there were not a sufficient number of existing integrated circuits, design tools, and trained designers.

In the same sense, some of the methodologies being explored in the RASSP program, such as top-down VHDL design with virtual prototyping, are not fully mature. Therefore, in assessing cost-effectiveness of these methodologies, the cost of developing infrastructure, such as simulation libraries, and providing training, should be distinguished from the cost of applying the methodology to a given application.

3. ARPA's Approach

The creation of a benchmarking component in the RASSP program represents a novel approach to assessing progress in a process-oriented development program. While process measurement and evaluation is not novel, there is no prior history from which to extract methodology or best practices for measuring improvement of a complex, evolving, design process such as RASSP. The approach adopted for benchmarking is driven by the challenges outlined above and a desire to address as many of the principal measurement and assessment goals as possible, subject to the resources available for both the execution and the evaluation of the benchmark. Wherever possible, existing measurement tools and process metrics have been chosen to exploit historical performance data, and proven measurement methodologies.

3.1 Benchmark Evaluation Process

Over the life of the RASSP program, the benchmark process involves concurrent activity in three major areas:

1. Developing benchmark applications and supporting material such as data, written specifications, executable requirements, test benches, and suitable performance and complexity metrics.

2. Evaluating benchmark execution including on-site meetings and observations, clarification and correction of requirements, milestone reviews, and interpretation of metrics.

3. Reporting, including benchmark evaluation reports, conference papers, and briefings.

3.2 Metrics

The principal RASSP performance metrics, such as design cycle time, are measured for a given benchmark application and compared to an estimate for standard practice to provide an indication of the relative performance of the RASSP process. However, coarse-grain metrics, such as design cycle time, provide little insight into where RASSP is improving or failing relative to standard practice. Therefore, for each principal metric, such as design cycle time, supporting metrics are necessary if insight into the benefits and deficiencies of the underlying process is desired. Table 1 provides a sampling of principal and supporting metrics.

Another use of metrics is to normalize application complexity across benchmarks. For a given benchmark, complexity is used to estimate the cost and design cycle time for a standard practice approach using parametric cost modeling. The observed cost and schedule required to complete a benchmark using the RASSP methodology is compared to the estimate of standard practice cost and schedule. Complexity metrics are also needed to assess the improvements in RASSP over successive benchmarks. In order to compare the cost and design cycle over successive benchmarks, normalized complexity numbers, such as source lines of code per person month (SLOC/PM), are used.

3.3 Standard Practice Model

As noted earlier, the principal RASSP metrics, such as design cycle time, depend on a large number of dependent and independent variables, and the scope of the benchmarking effort does not provide for the development of a complicated, parameterized model from first principles.

Three options were therefore considered for the current practice model:

1. A model based on average cycle times at each phase of the design could be developed for a representative application. However, an average model would not be capable of reflecting the specific conditions of a given benchmark, for example the level of experience of the staff, or the degree of reuse. Another problem with an average model is that it would not necessarily scale to the six-month duration of a benchmark.

2. A similar application might be identified and the actual cost and schedule compared against the performance on the RASSP benchmark. Several problems arise with this approach including locating a similar application, and accounting for the differences in personnel and technology at the time the similar application was developed.

3. A detailed parametric cost model could be developed for each benchmark. The model would be specialized to account for the specifics of the benchmark such as the number and experience of personnel, the complexity of the software and hardware, and numerous other variables that affect cost and schedule.

Table 1. *Principal* and Supporting Metrics

Metric	Typical Units
Design Cycle:	*Days*
•Process Step Time	Minutes/step
•S/W Reuse	NCSLOC[a] (%)
•H/W Reuse	Device (%)
•Productivity	NCSLOC/day
•Concurrency	--
•Delays	Days
Product Cost:	*Person-Hours*
•Components	Total Cost
•Manufacturing	Person-hours
•Testing	Person-hours
•Documentation	Person-hours
•MTTR	Hours
•Life Cycle	Est. Cost
Product Quality:	*Defects*
•H/W Defects	Defects/Unit
•S/W Defects	NCSLOC/Defect
•Time to Repair	Minutes
•System Defects	Defects/Release
•MTBF	Hours

[a] Non-commented source lines of code

The parametric model approach was selected based on cost-effectiveness, and adaptability to the specific character of each benchmark. Commercially available parametric cost estimation tools provide an estimate of the total cost and development cycle time to develop hardware or software. The parametric estimates rely on input parameters describing the hardware or software technology, the experience of the developers, and the complexity of the problem to develop a cost and development time estimate. An explicit model of the development process is not required, although some tools, such as the PRICE-S and SEER-SEM software estimators, allow mil-standard development processes. Parametric estimators account for the underlying process indirectly by calibration of the estimators through regression on historical data. Thus parametric cost estimation provides a mechanism for developing standard practice estimates for two of the three primary RASSP metrics, namely cost and design cycle time, without requiring an explicit current practice *process* model.

Parametric cost estimators also include a capability for estimating life cycle cost. Since production quantities of hardware are not planned as part of the benchmarks, and since the duration of the RASSP program is only four years, actual measurement of life cycle performance is not feasible. Estimation methods are therefore the only means available for assessing life cycle cost and supportability issues.

3.4 Parametric Cost Estimators

A key benchmark task is to measure the cost and design cycle time of the RASSP methodology relative to what would be achievable using industry standard practice at the start of the RASSP program in July of 1993. In order to accomplish this task, *multiple, commercial,* parametric cost estimation tools are employed for the following reasons:

1. The fact that a cost estimation tool is commercially available and supported suggests some measure of success has been achieved with the tool in accurately predicting cost and schedule.

2. Commercial tools incorporate "industry standard" default parameters many of which are updated on an annual basis. Dates in the past or future can be selected to account for the historical or projected impact of technology.

3. The use of more than one tool to predict cost and schedule helps ensure that the estimators are being used correctly.

4. The widespread availability and use of commercial cost estimation tools makes them a commonly encountered and well understood basis for representing the complexity of a particular application.

The PRICE family of parametric cost estimators, SEER, and REVIC, are all used in developing the standard practice baseline for a given benchmark [2].

4. Status

The first RASSP benchmark, which began in September of 1993, required the development of a VHDL-based virtual prototype for a SAR processor. The second benchmark, which is now underway, requires the Developers to carry the virtual prototype to a physical implementation as a means of assessing the fidelity and value of the virtual prototype model and methodology. At the conclusion of Benchmark-2, the actual time and cost to develop the SAR processor will be compared to an industry standard practice baseline. Expectations are that the effort expended in developing the virtual prototype will lead to significant reductions in the time and cost required to develop the processor hardware.

The VHDL modeling effort on Benchmark-1 has produced a limited database for calibrating parametric cost estimators for VHDL software development. Using lines of code as the principal metric, good agreement has been demonstrated between the parametric cost estimates and the actual VHDL effort. The calibrated parametric cost model can be used to estimate the savings associated with automatic code generation (synthesis of VHDL code), and reuse relative to manual coding of VHDL.

References

[1] B. W. Zuerndorfer and G. A. Shaw, “SAR Processing for RASSP Application,” *Proceedings of the First Annual Rassp Conference*, August, 1994.

[2] J.C. Anderson, “Predicting the Future with RASSP Benchmarks.” Proceedings of the First Annual RASSP Conference, August, 1994.

Rapid Prototyping of Application Specific Signal Processors: Current Practice, Challenges, and Roadmap

by Vijay Madisetti and Jack Corley

The authors present a “current practice (circa 1993)" model for the design and prototyping of application specific (e.g., signal processing) parallel processors. A number of limitations in current design practice are highlighted together with challenges faced and a roadmap for candidate solutions. The Rapid Prototyping of Application Specific Signal Processors (RASSP) project of the US Department of Defense (ARPA and Tri-Services) targets a 4X improvement in the design, prototyping, manufacturing, and support processes (relative to current practice).

1. Introduction

This section introduces classes of high-performance application specific parallel processors. Section 2 presents “current practice (circa 1993)" based on extensive study of industrial practice through first-hand communications with various industrial and defense contractors at Lockheed Sanders, Martin Marietta, Hughes Avionics Systems, Motorola, MIT Lincoln Laboratories, and Raytheon corporations, by the authors as part of the RASSP Education and Facilitation effort over the past year. Section 3 describes the challenges facing application specific parallel processor specification, implementation, verification and support, and section 4 outlines the areas of focus of the RASSP [1] efforts that attempt to improve upon the current practice in a manner expected to have a lasting impact on the way signal processors are procured, designed, and manufactured [2].

1.1 Representative Architecture

A typical high-performance avionics parallel signal processor operation flow is shown in Figure 1 [3]. The inputs are recorded by sensor arrays, and the data is pre-processed by an array of (typically hardwired) computational elements, the *Sensor-Specific Processing (SSP)*, that are optimized with the sensor array and the recording environment. Typical SSP operations include range adjustment, background subtraction, matched filtering, and track-to-track correlation. Given the high computational throughput, restricted functionality and severe form constraints (size, volume, area and power), the SSP functions are typically ASICs with non-standard interfaces.

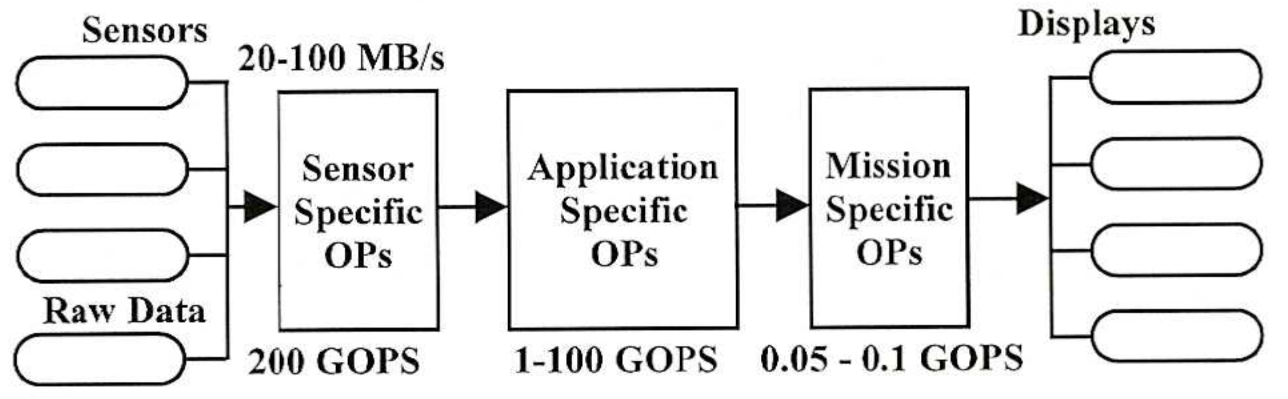


Figure 1. Typical System Architecture

After this time-critical processing is completed, the *Application Specific Parallel Processing (ASPP)* (about 30-100 processors) is commenced on an array of processors and communications elements, with appropriate test, control, and maintenance structures. Typical ASPP operations include coordinate transformation, Kalman filtering, tracking, and parametric estimation and involve application related functionality. The ASPP functions also require relatively high throughput, and should have as much flexibility (i.e., programmability) as possible. In an ASPP implementation lies the problem of multi-objective function optimization and tradeoffs among form factors, performance, programmability, ease of upgrades, and capability for test and diagnostics.

The *Mission-Specific Processing (MSP)* typically requires interpretation of the ASPP processing, and can be confined to a few processors that are often co-located within the ASP box. These functions include clutter analysis, track handoff, decision analysis, kill assessment, etc.. The relative orders of processing in GOPS are also depicted (decreasing to the right) in Figure 1, while memory requirements increase towards the right, with the most memory required by the MSP stage (typically 200-400 Mbytes). Typical high-end form factor constraints for volume, power, weight, and I/O rates are in the order of 2-3 ft^3, 40-500W, 10-60 lbs, and 30 Mbytes/second, while for low-end low power portable applications they are considerably more severe (in size and power). Interprocessor communication bandwidth requirements can range between 40-1000 Mbytes/second.

1.2 Architectural Design Space

A combinatorially significant number of alternatives exist in the implementation of the SSP, ASPP, and MSP functionality. A few key architectural attributes [3] are listed below:

- **Computational Elements** -- types (data, control, ASICs, or DSP) of processors, coarse or fine-grain task assignment, heterogeneous or homogeneous processing, size of memory elements, degree of coupling between memory and processors, software and algorithms utilized, SIMD or MIMD types of control.

- **Communication Elements** -- buses used, backplane architectures, interface to buses within system, interfaces to environment, routing and communications protocols.

- **Topologies** -- allocation of physical resources (processors, memories, communication elements), interconnection topologies and technologies (fiber, parallel/serial, etc.), I/O configurations, integrated test and fault-tolerance capabilities.

1.3 Typical System Specifications

A number of specifications/requirements form the input to the prototyping process, and they can be weighted in relative order of importance. These include (in no specific order):

(1) Functionality and Performance,
(2) Environment,
(3) Interfaces and Packaging,
(4) Security,
(5) Schedule for deployment,
(6) Cost,
(7) Software and Hardware restrictions,
(8) Size and Volume,
(9) Weight,
(10) Power,
(11) Reliability,
(12) Maintainability,
(13) Fault-tolerance,
(14) Scalability,
(15) Standardization.

A successful design has to achieve a satisfactory degree of compliance with each of these specifications [3].

2. Current Prototyping Practice

We now examine current practice (1993) in the prototyping of application specific parallel signal processors.

2.1 System Development Phases

The life cycle of a typical large system roughly follows the six phases shown in Figure 2. The focus of this study is on phases 3-6. Often, phase 3 is bypassed, and the sequence shown by the dotted-line is followed. In some cases, phase 3 follows 4, and is followed by phases 5 and 6. The dollar cost figures for each of these phases in terms of resources (capital and personnel) can run into a few tens of millions in the initial phases of Figure 2, and as high as a few hundreds of millions (or more) in the later phases of system development, deployment and maintenance. The cost incurred/committed rises steeply with the onset of phase 3.

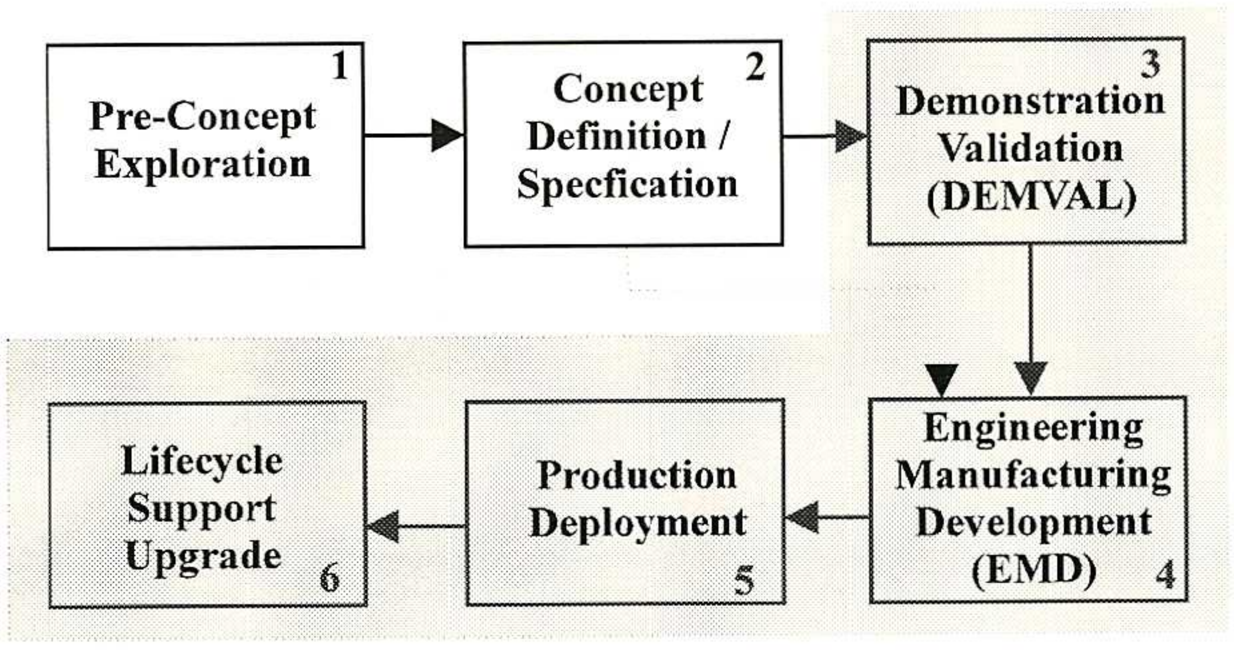

Pre-Concept Exploration 1
Concept Definition / Specfication 2
Demonstration Validation (DEMVAL) 3
Engineering Manufacturing Development (EMD) 4
Production Deployment 5
Lifecycle Support Upgrade 6

Figure 2. Typical System Development Phase

Clearly, decisions and tradeoffs made in the early phases have significant financial impact later in the system development lifespan. In addition, the long time span (often 8-10 years) between phases 2 and 5 render some aspects of the technology obsolete by the time a concept is fielded. While customer input is significant in phases 1-3, it diminishes in phases 4-6 leading to

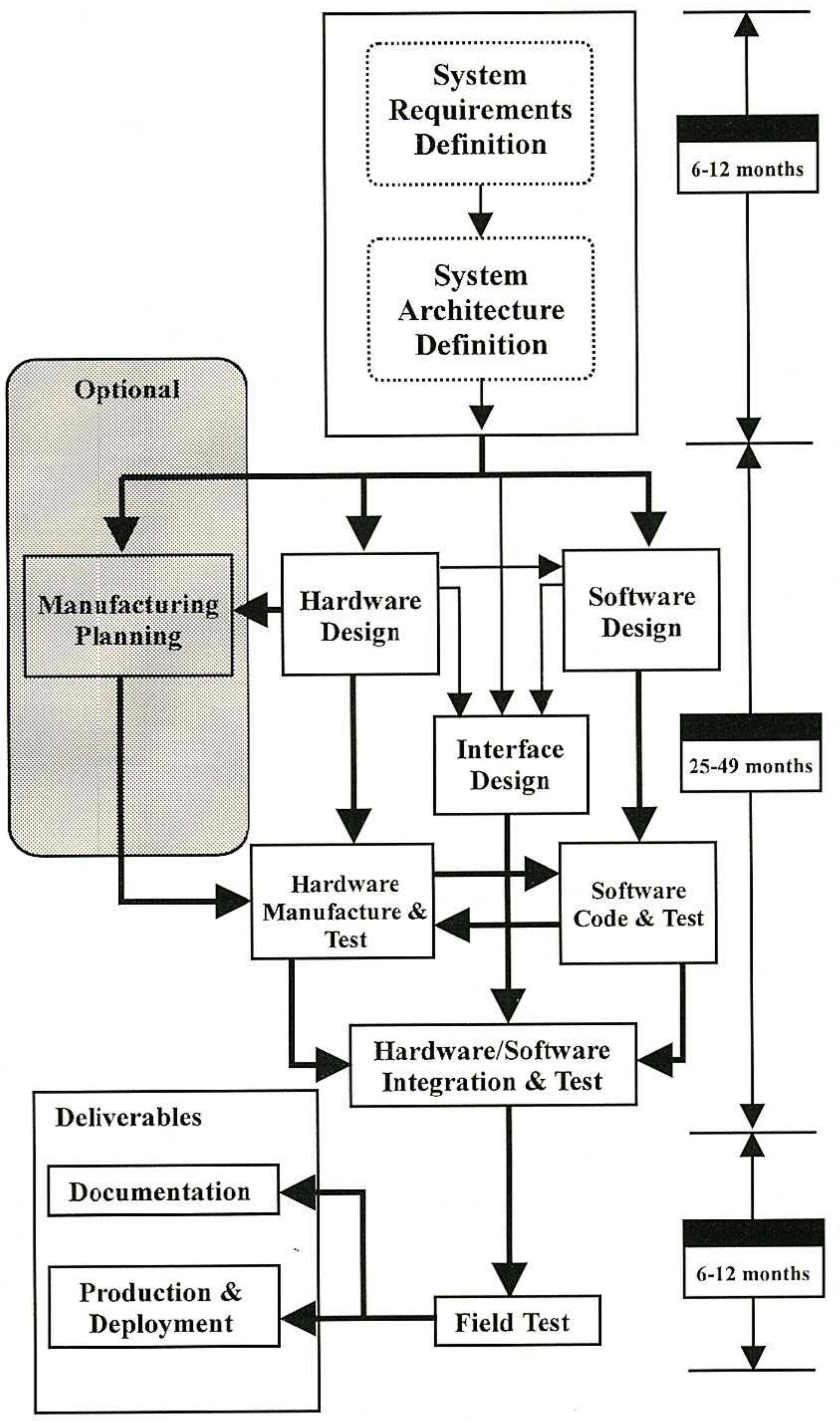


Figure 3. Current Practice Design Process (1993)

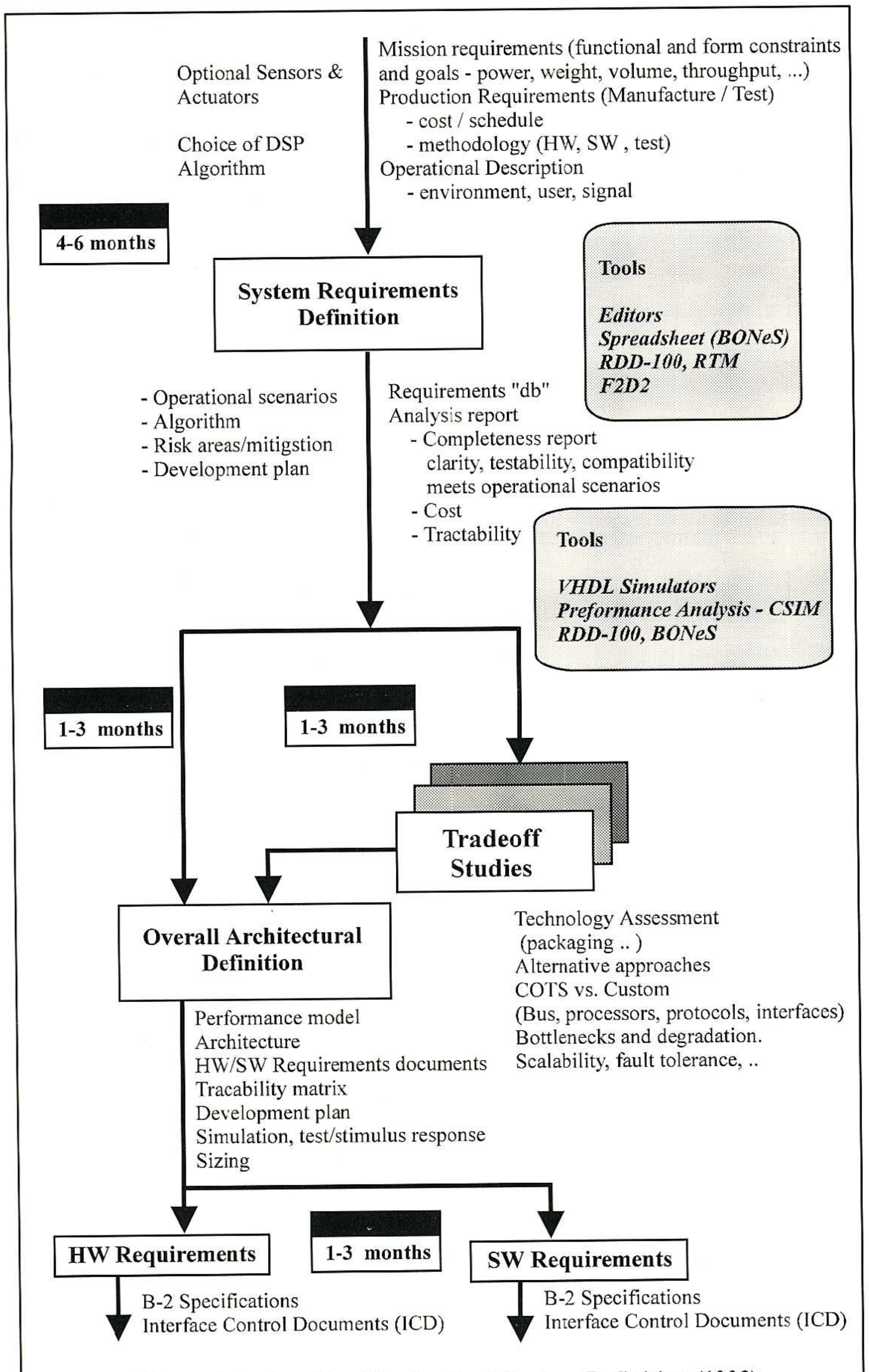

Optional Sensors & Actuators
Choice of DSP Algorithm
Mission requirements (functional and form constraints and goals - power, weight, volume, throughput, ...)
Production Requirements (Manufacture / Test)
- cost / schedule
- methodology (HW, SW , test)
Operational Description
- environment, user, signal
4-6 months
System Requirements Definition
Tools
Editors
Spreadsheet (BONeS)
RDD-100, RTM
F2D2
- Operational scenarios
- Algorithm
- Risk areas/mitigstion
- Development plan
Requirements "db"
Analysis report
- Completeness report
clarity, testability, compatibility
meets operational scenarios
- Cost
- Tractability
Tools
VHDL Simulators
Preformance Analysis - CSIM
RDD-100, BONeS
1-3 months
1-3 months
Tradeoff Studies
Overall Architectural Definition
Technology Assessment
(packaging ..)
Alternative approaches
COTS vs. Custom
(Bus, processors, protocols, interfaces)
Bottlenecks and degradation.
Scalability, fault tolerance, ..
Performance model
Architecture
HW/SW Requirements documents
Tracability matrix
Development plan
Simulation, test/stimulus response
Sizing
HW Requirements
1-3 months
SW Requirements
B-2 Specifications
Interface Control Documents (ICD)
B-2 Specifications
Interface Control Documents (ICD)

Figure 4. System Specification/Architecture Definition (1993)

significant risks to the manufacturer in fixed-price contracts, or to the customer via cost overruns when redesign or rework is required. Life cycle support (phase 6) can last between 10-30 years. The capability for rapid upgrade is an effective insurance against technological obsolescence of the fielded system [6].

2.2 Current Practice (Circa 1993)

Figure 3 presents the current practice model for system design. The various stages in a "waterfall"-type process flow are demarcated together with time ranges (min, max) for each stage. Figures 4, 5, 6, and 7 describe each of the stages of Figure 3 in greater detail, again presenting details of the time required for each substage, together with tools used, process inputs and outputs. Because the figures are very detailed, it is hoped that they are self-explanatory. The time lines have also been validated via communications with the industrial entities involved in large system design and implementations.

3. Areas for Improvement

Some observations with respect to the design flow in Figure 3 that provide pointers to areas for improvement [6] are described next.

3.1 Automation and Enterprise Integration -- Though progress in automation has been significant in the area of Hardware Design, system-level design, architectural exploration and tradeoffs, far less progress has been made in software design, hardware/ software integration, or in integrating manufacturing and product design activities and database libraries. Most of the information transferred between various stages of the design process is manual and is usually documented on paper with little standardization. The price (in cost and time) for this lack of integration is paid by the system designer who has to manually translate descriptions from one CAD tool to another with little, if any, interoperability provided (this is time-consuming and expensive considering that over thirty individual point tools are used at various stages in the

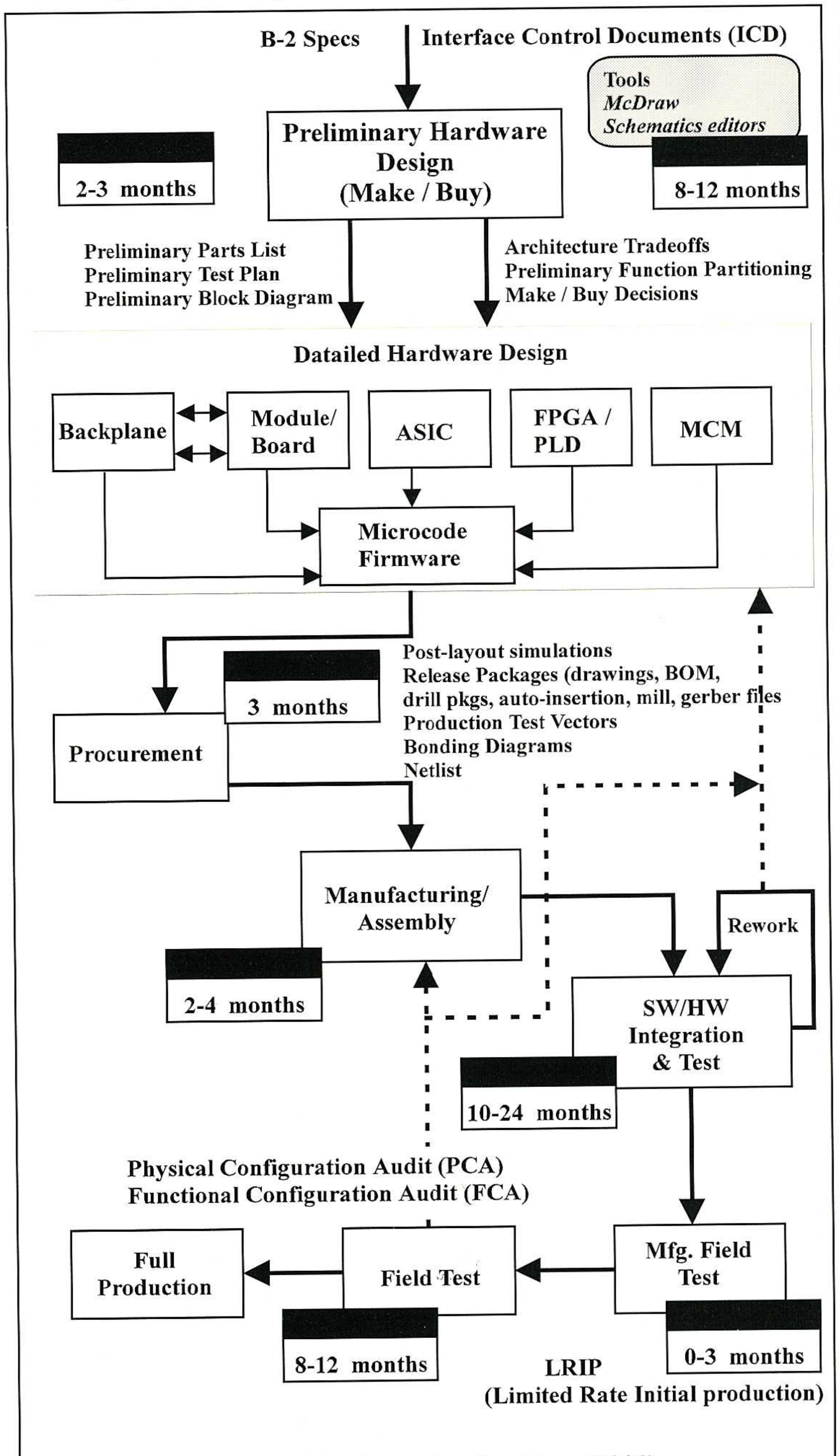


Figure 5. Hardware Design Flow (1993)

design process flow of Figure 3). Standardization efforts have been initiated very recently in an effort to meet these needs through the use of VHDL [7].

3.2 Design with Reuse -- Application specific systems can benefit from reuse of design information from past designs. Algorithms can be rapidly designed using reuse libraries of commonly used functional blocks. Architectures can be quickly synthesized using reuse components from past designs. Thus reuse is a feature that can be leveraged with advantage in cutting down the prototyping times incurred in large projects, if there were a mechanism to formally capture reuse information in a form that could rapidly be assimilated in an application specific design. Figure 3 provides little opportunity for design with reuse.

3.3 Design for Reuse -- In continuation of the previous point, any successful attempt at resolving the prototyping bottleneck must include a methodology to ensure that future designs can benefit from current design efforts. This can be facilitated if efforts were taken to populate libraries of reuseable components (from the current projects) in a form that the future design efforts can reuse. Contrasting with Design with Reuse which takes place "in-cycle," Design for Reuse can be initiated "off-cycle" via population, maintenance, and upgrades of VHDL reuse libraries.

3.4 VHDL-based Co-Development and Codesign Methodologies and Virtual Prototyping -- True HW/SW codesign allows both hardware and software to be designed within a common framework and simulated together before being fabricated. Current practice attempts to automate this process via HW/SW Interface partitioning followed by three individual paths to HW, SW and Interface design and implementation, respectively. A drawback with this approach is that (as shown in Figure 8(i)) software can be designed and tested only if the hardware platform (at board and rack levels) is available. The latter is time- and cost-consuming. Software is not just application specific software, but also control, diagnostic and test software. Often, control, diagnostic, and test software requires an order of magnitude larger man-hour effort to develop than does application software [6]. Conventional hardware software codesign methods assign a token interest in the issue of software required for control, diagnostic and test purposes, and attempt to catch all integration issues under the term "interface." The approach shown in Figure 8(ii) represents a "true" HW/SW codesign wherein software models (in VHDL) of HW are provided to the SW developers and the entire software is designed, tested and integrated with the HW models long before any hardware is fabricated or manufactured. Thus, the design loops L_1 and L_2 are quick and require no hardware fabrication & engineering cost and, in addition, provide capability for complete system design using a process known as *virtual prototyping* [4]. The assumption, of course, is that libraries of HW models in software are available. VHDL can be used with advantage in this true HW/SW codesign philosophy -- one that embraces a hardware-less system design. Our experience has shown that hardware-less HW/SW codesign is very efficient, reduces time for HW/SW integration to a matter of weeks and also allows rapid upgrades, together with significant savings in cost. Once virtual prototyping is completed, the field prototype can be quickly and efficiently manufactured.

3.5 Executable Requirements and Specifications Figure 3 highlights the fact that current practice is to provide processor and

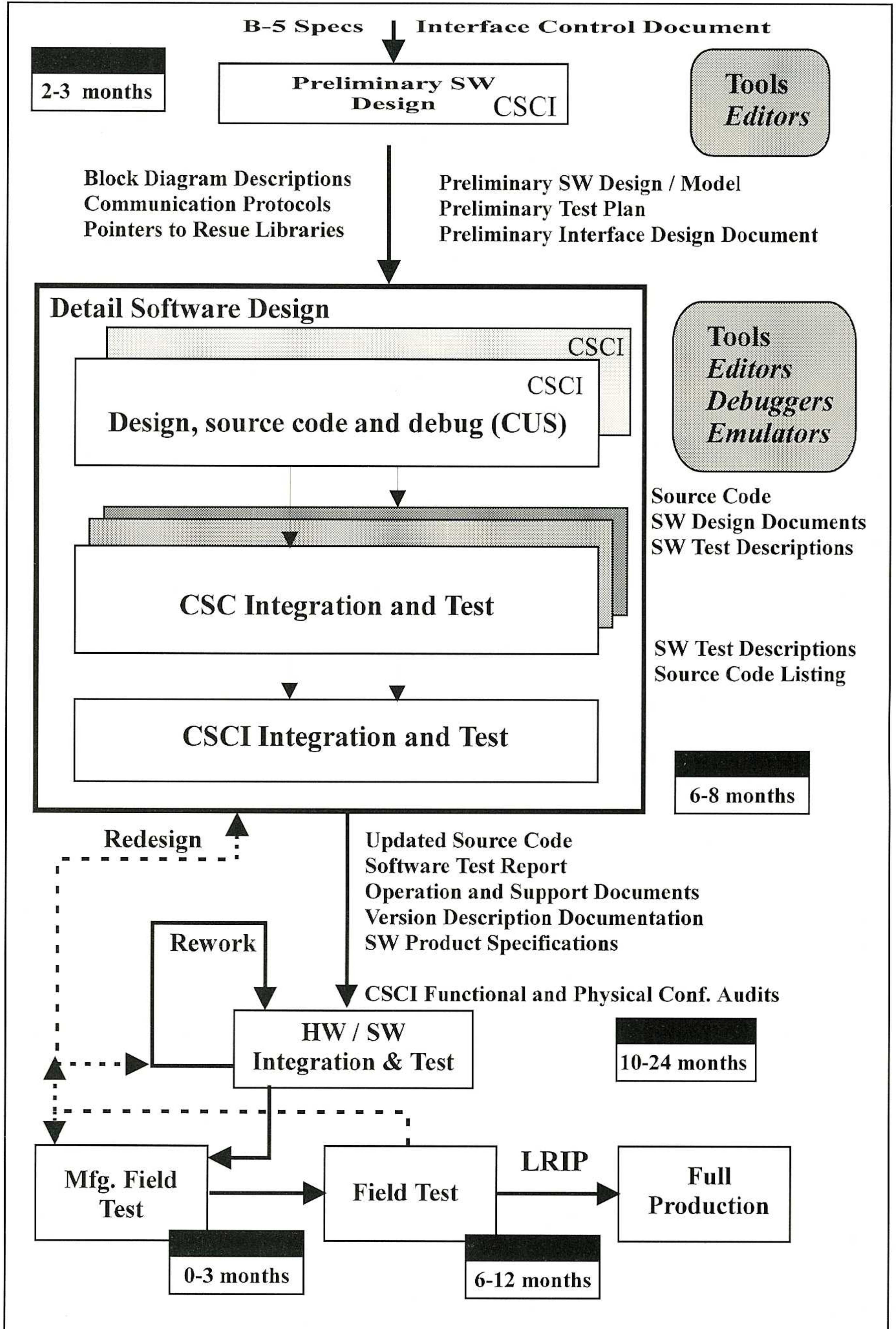


Figure 6. Software Design Flow (1993)

system requirements in written form, often in hundreds of pages of documentation which must be interpreted and captured by a requirements traceability tool. Requirements that are machine readable and executable are invaluable both in terms of efficiency and accuracy of interpretation as well as in regression testing of the design over many abstraction levels. An executable requirement of a complex application specific system would be an effort that pays for itself in later stages of the design process. The final design itself can be documented in the form of an executable specification which would be machine readable and capable of being executed. Executable specifications simplify later upgrades of legacy system and also in reducing life-cycle costs. MIT-Lincoln Laboratories has initiated some work in this area as part of the RASSP effort.

3.6 Modular Software and Hardware Development -- VHDL-based software and hardware development supports HW/SW codesign and, in addition, enables reuse of application specific components. Structured system design and development environments such as that shown in Figure 8(ii) utilize effective software engineering principles, significantly reduce the loss in design quality and requirements traceability while passing from one stage to another in the current practice model of Figure 3 and are a requirement for any new design methodology for rapid prototyping. The notion of a *standardized virtual interface (SVI)* between all constituent hardware and software components to ensure rapid "plug-and-play" capability would be attractive for rapid prototyping, and its implications in terms of standardization and efficiency remain to be explored [5].

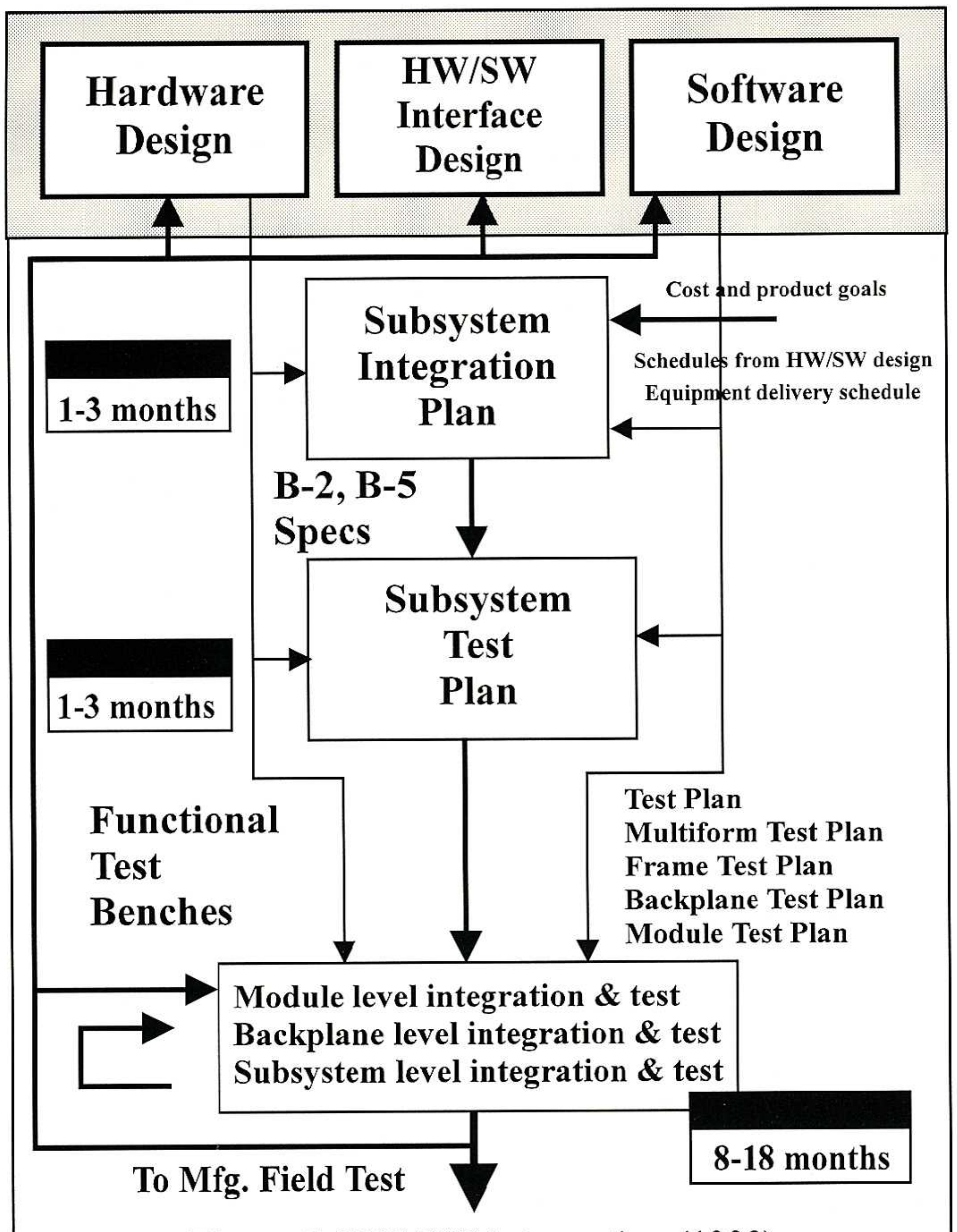
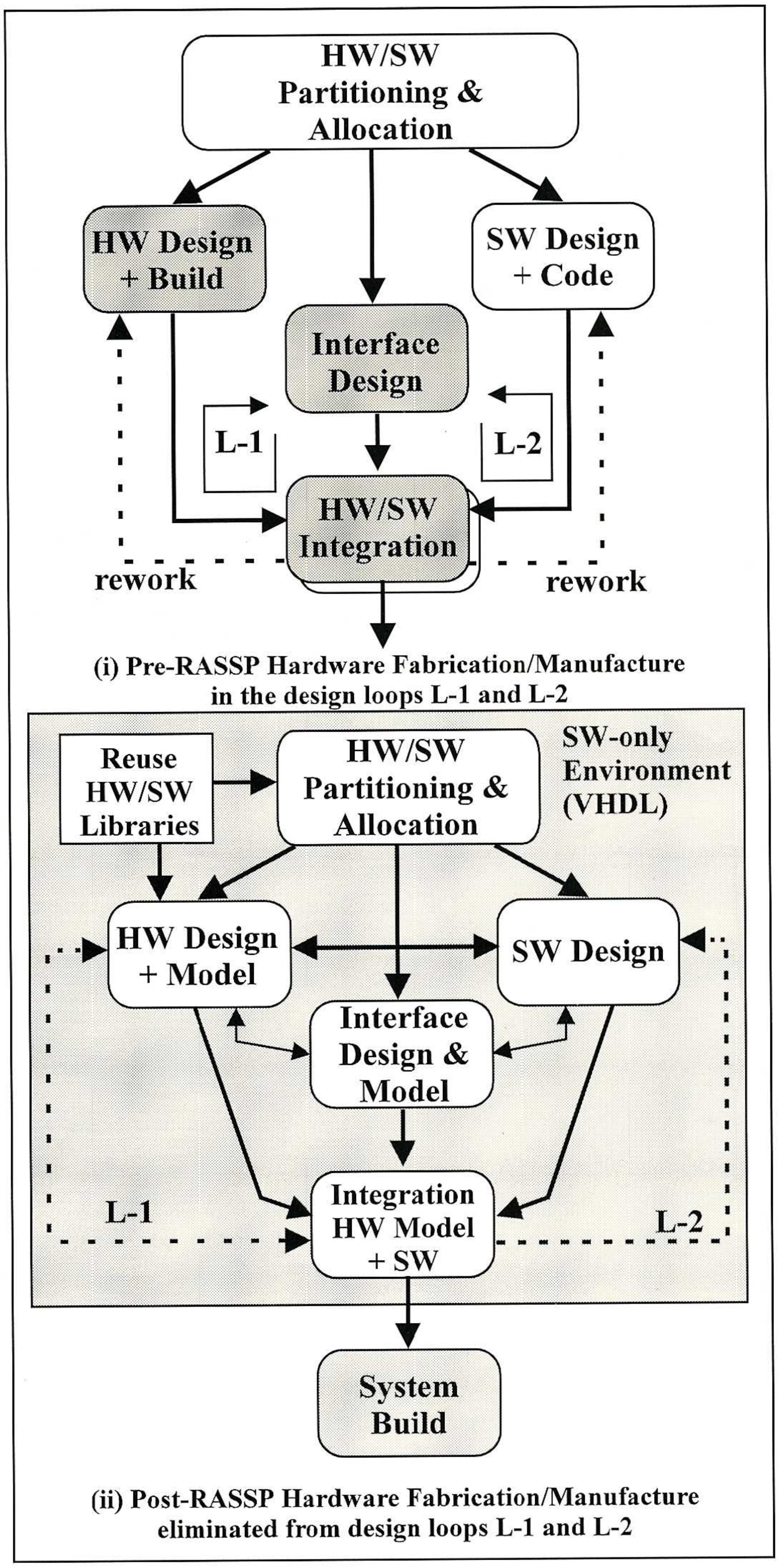


Figure 7. HW/SW Integration (1993)



(i) Pre-RASSP Hardware Fabrication/Manufacture in the design loops L-1 and L-2

(ii) Post-RASSP Hardware Fabrication/Manufacture eliminated from design loops L-1 and L-2

Figure 8. HW/SW Codesign

(i) Current practice (1993-1994)
(ii) Post-RASSP
Note elimination of hardware fabrication, assembly and board/system level manufacture from the design loops. Savings result in time and cost, and the capability for customer input and concurrent life-cyle support and upgrade is enhanced. Shaded areas imply hardware.

3.7 Integrated Process and Product Development Teaming

Integrated manufacturing, product and design teams provide the tight linkage required to efficiently and concurrently create new products quickly. Enterprise integration allows this tight coupling between hardware and software design, manufacturing procurement, reliability, maintainability, and supportability when utilized in conjunction with a top-down design methodology. The current practice model of Figure 3 illustrates recent efforts in this direction, though room exists for further integration.

4. Summary

For the first time a detailed picture of the current practice (1993) design flow in the design, prototyping and deployment of high performance application specific parallel processing systems is presented. Some limitations of the current practice approach are outlined, and improvements sought by the US Department of Defense's RASSP program are presented.

Acknowledgments

The authors acknowledge with gratitude inputs from various RASSP participants and programs. However, the views expressed in this paper are the authors' own and do not necessarily represent the viewpoints of the US Department of Defense, ARPA, MIT-Lincoln Laboratories, Lockheed Martin-ATL, Lockheed Martin- Sanders, Raytheon, Hughes, Motorola, or SCRA.

References

[1] M. Richards, "The Rapid Prototyping of Application Specific Signal Processors Program," Proc. of *First Annual RASSP Conference*, August 1994.

[2] L. Scanlan, "RASSP: Road to 4X," *The RASSP Digest*, Issue 2, Vol 2, 2nd Qtr 1995, (http://rassp.scra.org).

[3] F. Shirley, "The RASSP Architecture Guide - Rev. C," Document AVY-L-S-00081-101-C, Lockheed Sanders Inc., Nashua, NH, April 14, 1995.

[4] V. Madisetti, T. Egolf, S. Famorzadeh, L-R. Dung, "Virtual Prototyping of Embedded DSP Systems," Proc. of *IEEE ICASSP 95*.

[5] G. Caracciolo, J. Pridmore, "Architectures for Rapid Prototyping of Embedded Signal Processors," Proc. of *IEEE ICASSP 95*.

[6] G. Shaw, V. Madisetti, "Assessing and Improving Current Practice in the Design of Application Specific Signal Processors," Proc. of *IEEE ICASSP 95*.

[7] Proceedings *VHDL International Users' Forum (VIUF)*, Spring 1995.

Timing Insensitive Binary-to-Binary Translation (TIBBIT)

by Bryce Cogswell and Zary Segall

1. Introduction

The TIBBIT project is developing novel methods of applying binary-to-binary translation (BBT) technology to real-time and embedded applications. While BBT has been used commercially for translating workstation-class applications by companies such as DEC, HP, Tandem, NonStop and others, the technology has yet to be applied to the domain of real-time and embedded systems in which applications can be very tightly tied to the underlying hardware in terms of both functionality and timing. Translation methods developed under TIBBIT allow multiple real-time and embedded applications to be migrated from dedicated processors to newer, faster multiprocessing systems while ensuring that the hardware/software interfaces, and the timing of I/O events generated or processed by the applications, is kept equivalent to that of the original application and platform. A retargetable framework is employed which supports a wide variety of architectures including digital signal processors.

In the pursuit of increased performance at reduced cost, real-time and embedded applications may resort to *ad hoc* methods of enforcing the timing of operations, and there is little guarantee that event timing will be regulated by hardware alarms or a timer-driven scheduler. The result is that migration to a different processor with different timing can disrupt the careful balance of timing incorporated in the original design, leading to subtle bugs or complete failure on the target platform. Figure 1 diagrams the problem of ensuring equivalence of both application *and* hardware interactions.

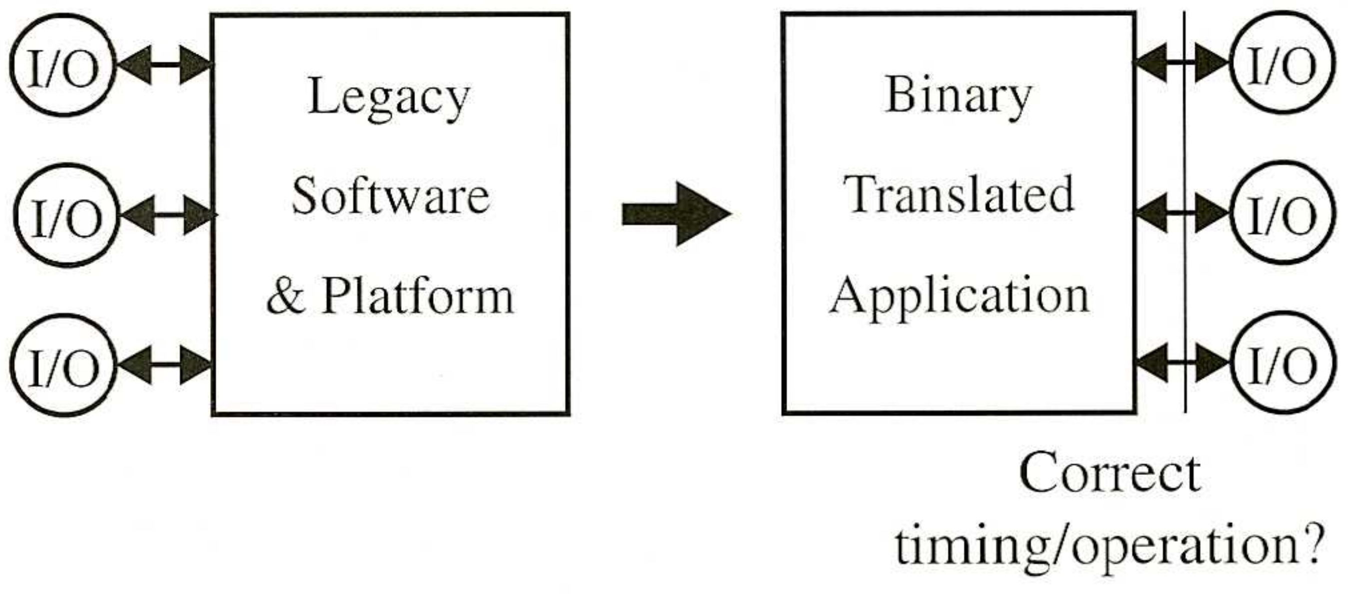


Figure 1. Problem of maintaining correctness of I/O accesses.

The goals of the TIBBIT methodology are as follows:

- *Semantic equivalence*: The resultant program is semantically equivalent to the original program.

- *External timing equivalence*: The timing of the program on the target is equivalent to the source platform within some predictable error bound.
- *Processor independence*: The scheme should be effective across a wide range of processor architectures.
- *Use existent I/O architecture*: The interfaces to external devices to which the source processor is attached are preserved.
- *Quantifiable performance*: The success of a translation can be predicted prior to translation, and the degree of timing equivalence can be quantified.
- *Automated translation*: The translation process should be nearly or entirely automated.

2. Approach

The type of code that is of concern when migrating to a faster processor is that which implicitly uses the processor performance to regulate program timing, such as:

```
Read_Port();
...compute...
Read_Port();
...compute...
Read_Port();
...compute...
```

The minimum delay between consecutive reads of the port, which is satisfied on the source processor by the time spent performing intermediate computations, may or may not be satisfied after migration to a faster target.

Our approach to the problem is to precisely track and mimic the timing behavior of the code as it was when executing on the source processor. During the translation from the source to the target, a timing code is inserted into each block of instructions that describes the amount of time required to execute that block on the source processor. This timing information is analyzed as the target processor executes each block in turn, and is used to compute the time at which the current block on the target would be executed on the source. This forms a virtual clock which tracks time as it passes on the source. We call this the *source clock*. It contrasts with the *target clock*, which is the real, or wall-clock, time. Figure 2 shows how the code is augmented with timing information such that even in the presence of loops or conditional execution, the time spent executing the block on the source processor is known.

At regular intervals during execution the target processor compares the values of the source clock and the target clock and determines the difference between its progress and the progress the application would have achieved on the source processor. The result is used as feedback to the scheduler that runs the various applications that have been migrated to the target processor.

Scheduling on the target can be done two ways: To maximize the degree of timing equivalence, a dedicated target processor can be used; the application can be scheduled on the target under rate monotonic scheduling. Using RMS allows multiple TIBBIT translated and native applications to be executed concurrently.

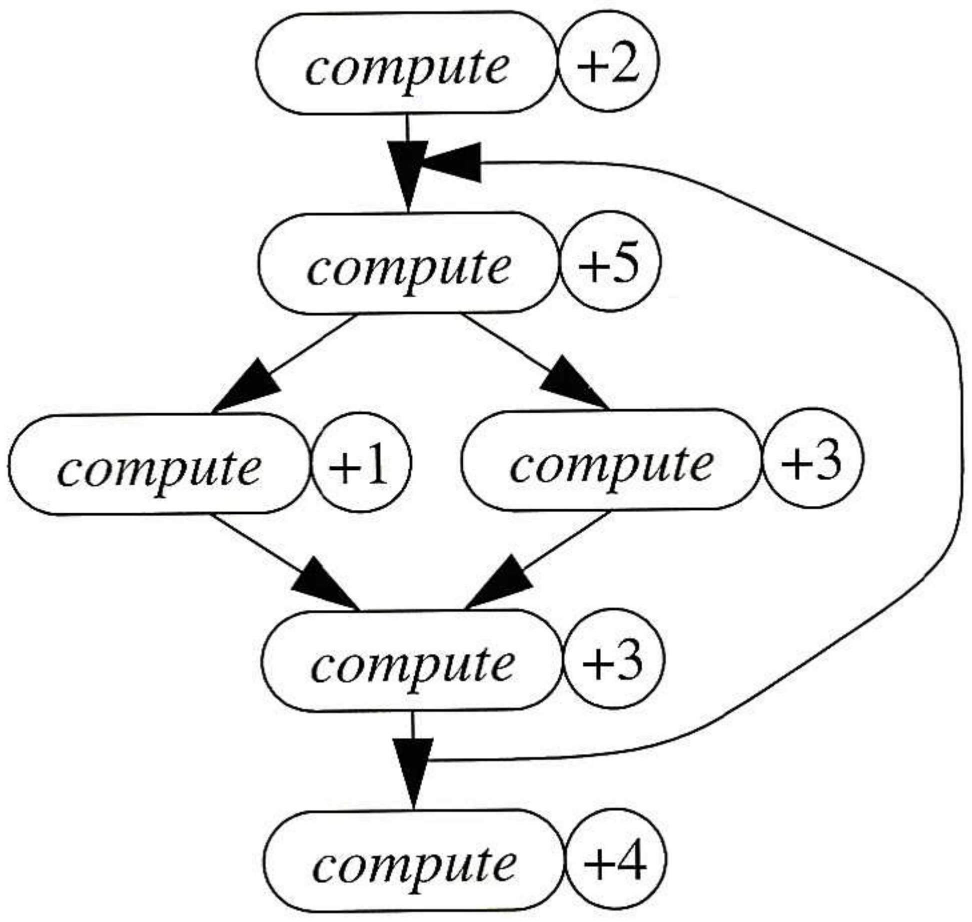


Figure 2. Code fragment augmented with timing information.

3. Analysis

Given an application that a user wishes to binary translate to a target platform with a specified degree of timing equivalence, we wish to determine whether the translated application will meet the user-specified timing-equivalence requirements under worse-case conditions. The modeling of translated tasks is performed by considering the maximum amount of time required on the target to execute a code fragment requiring a given amount of time on the source processor, and then adding in the overhead of performing the TIBBIT scheduling.

Table 1 provides a summary of the parameters impacting TIBBIT schedulability, while Table 2 specifies the greatest amount by which timing on the target processor can become out of sync with the source. The precondition column specifies the condition that must hold for the task to be schedulable under TIBBIT, while the max behind and max ahead columns bound the maximum timing error that can occur.

Symbol	Definition
T_d	A user-selected time interval on the source.
t_{ov}	Maximum time for target to execute code requiring time T_d on source.
T_c	TIBBIT instrumentation granularity.
t_{clk}	Time to read real-time clock on target.
t_{csw}	Scheduler overhead for target.

Table 1. Summary of TIBBIT model parameters.

Target processor	Precondition	Max behind	Max ahead
Dedicated	$T_d \geq t_{ov}+2t_{clk}$	$t_{ov}+t_{clk}$	T_c
Multitasking	$T_d \geq t_{ov}+2t_{csw}$	T_d-t_{csw}	T_d+T_c

Table 2. Summary of TIBBIT algorithm characteristics.

4. Results

The algorithms and models developed under TIBBIT have been validated by translating a variety of applications developed for the Motorola M68000 Education computer to both Unix and PC platforms. Most of the applications are drawn from a mix of C and assembly language programs given as lab assignments for the undergraduate Real-Time Systems class at Carnegie-Mellon University, and represent systems whose implementation and timing is unknown to the user of the translator.

The timing equivalence of translated code has been measured as accurate as 80 microseconds in the short-term and 0.1% in the long term, with an overhead of about 20% additional processor utilization due to timing instrumentation.

An example of the abilities of the system is an application that reads a data set from the serial port, performs a simulation based on the data, and returns a single byte indicating whether the simulation was successful or not. A host program executes on another machine, communicating via the serial line, and records the time recorded for the simulation to complete for various data sets. This test highlights the problem of modeling the timing behavior of an application whose I/O timing is completely data dependent. The time required for each simulation varies according to the contents of the data set, and it is essentially impossible to determine how long it will last before performing it.

Figure 3 shows the execution time recorded by the host for a particular data set, in which the RMS period the translated application is scheduled with is varied from 100 to 100000 microseconds. 10 trials are run at each period, and the graph shows the average and the actual measured values. The 'V' shows the bounds on timing that are predicted by the model, while the line in the center gives the average timing of the trials.

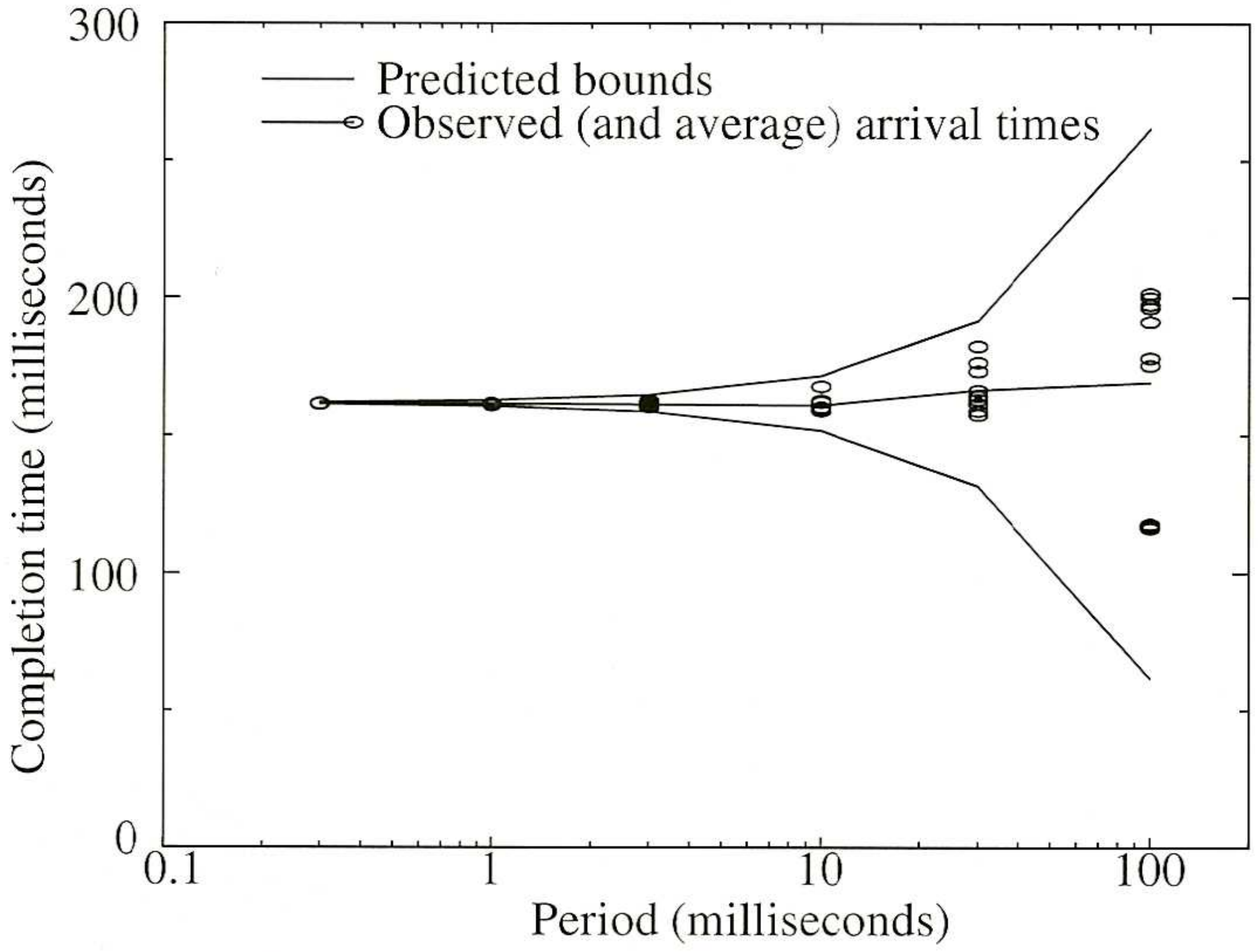


Figure 3. Timing error as a function of period.

Increasing the period decreases the processor utilization required on the target, since context switching and scheduling overhead is reduced. At the same time, however, the accuracy of the time equivalence is diminished. The graph demonstrates that the bounds predicted by the models are accurate over a range of scheduling periods.

The largest application translated to date is a real-time M68000-based operating system, implemented in a combination of C and assembly code, which allows one to execute one of 6 different applications embedded in it, each application spawning from 1 to 8 tasks that interact with each other, the O.S. and the user. The operating system contains a flexible scheduler that can be configured to use any of the common scheduling algorithms and provides support for semaphores, priority control, interrupt driven or polled I/O, etc.

This application was assigned as the final project of the Real-Time Systems class, and the applications that execute on top of it are designed to stress the system and make evident weaknesses in the performance and fairness of the real-time scheduler. TIBBIT translation of this application similarly stresses the ability of the target platform to maintain those same timing constraints, and when the degree of timing equivalence is relaxed beyond about 5 millisecond, many of the sample applications fail with obvious regularity.

5. Conclusions

The TIBBIT project has developed a means of binary translating real-time and embedded applications from slower to faster processors while maintaining the timing characteristics of the original host. The translation is performed by augmenting the translated code with timing information from the source processor, and using that information at run time to ensure that the timing of events corresponds with the original timing. The algorithms have been modeled and validated under a real implementation, and results demonstrate that timing equivalence can be maintained to within 80 microseconds or 0.1%.

Graph Translation Tool (GrTT)

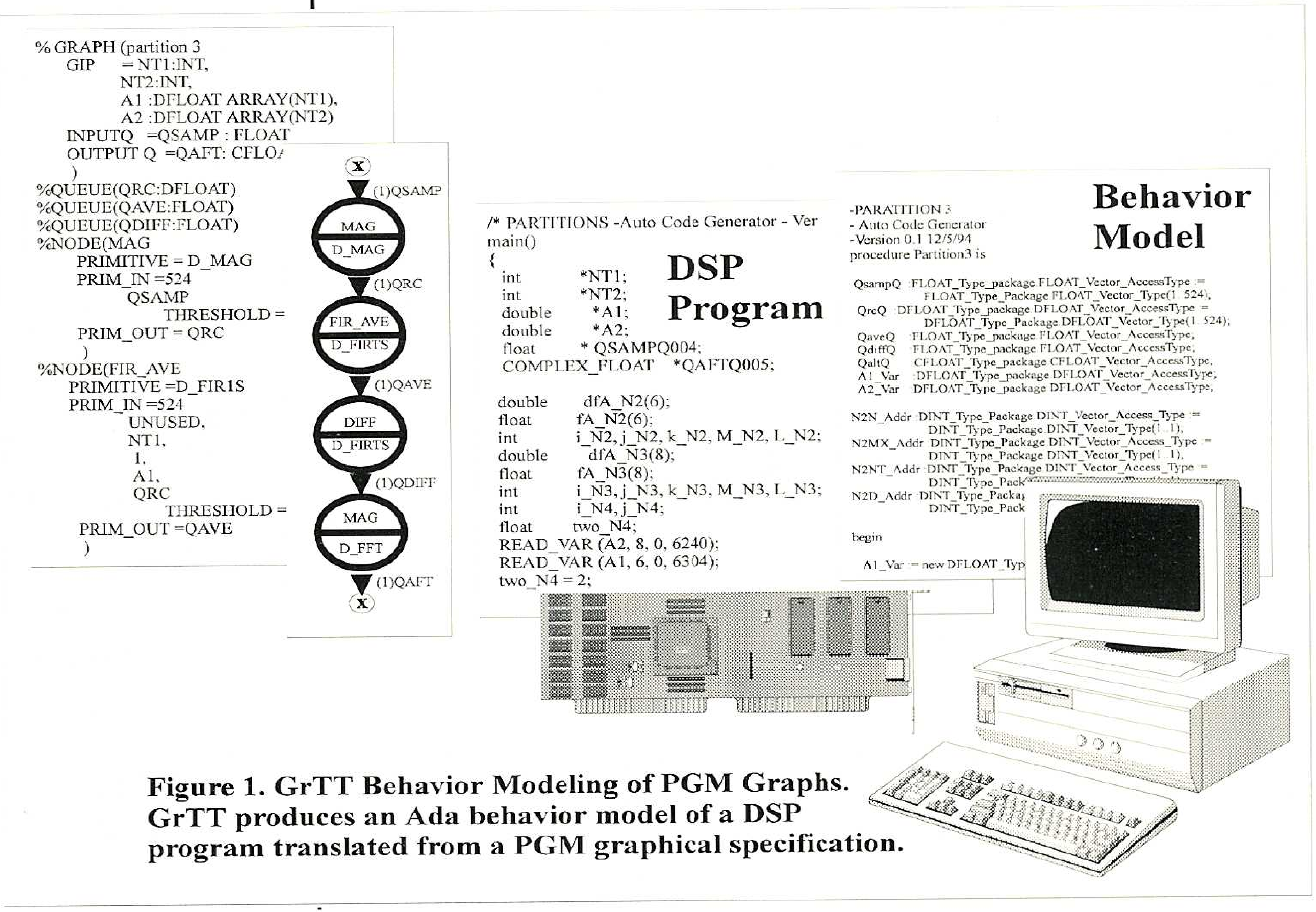
by Chris Robbins

Management Communications and Control, Inc. (MCCI) is developing GrTT (Graph Translation Tool) under a RASSP technical base BAA contract. GrTT is an autocoding tool that will translate Processing Graph Method (PGM) graphs to Ada behavior models. GrTT may be used to create behavior models of either hardware or software architecture partitions of PGM data flow graphical application specifications. The functional behavior of the model will be identical to the graph partition represented. Identical outputs will be produced by either model execution or data flow execution of the processing graph on a common input data set. A dynamic view of model execution is supported thus providing visibility of the modeled graph's execution behavior.

Implementation of the RASSP HW/SW codesign process by the Lockheed Martin Advanced Technology Laboratories Team utilizes PGM for data flow specification of the application. Processing within the PGM graph's nodes is specified by domain primitives. Domain primitives are target independent signal processing and data flow control function specifications. Their use in the PGM application specification provides an open application programmer's interface (API) to the team's tools implementing the architecture selection and design processes. Domain primitive graphs are partitioned by the architecture tools into hardware and software allocations. The allocations are further partitioned to become either hardware component partitions or software partitions. Software design tools will generate stand-alone PGM graphs for each partition. GrTT may be used to generate behavior models for each hardware or software partition.

Figure 1 illustrates the partition modeling concept. An application partition graph is shown on the left in both iconic and notational form. Each node has its unique name above the line and specifies the domain primitive implementing the node below the line. Queues represent FIFO buffering of the data between the nodes. Node execution parameters associated with the node ports that are linked to queues specify a thresholding criteria for node execution, data amounts to be read, and data amounts to be consumed from the queues upon node execution. Data amounts produced onto output queues per node execution are functions of the domain primitive controls, read amounts, data modes, etc. Node execution parameters, process controls and parameters may be made run-time variables and provide the capability to externally modify graph execution. Data flow execution (execution of nodes when thresholding criteria are met) guarantees determinism or causal behavior of the graph. GrTT accepts application partition graphs in their notational form plus sets of enumerated values of graph variables, and it produces an Ada procedure that is the behavioral equivalent of the input graph. Graph variables that cause the input graph to alter data flow, node firing rates, or primitive processing will cause identical behavior in the behavior model that GrTT produces.

GrTT consists of three major objects: the SPGN parser, graph analysis, and autocoder objects. GrTT is supported by the domain primitive database which provides data support for both GrTT, the target independent translation, and the RASSP target dependent translations of domain primitive graphs. The translation process that these objects implement is illustrated in Figure 2. The SPGN parser accepts a partition graph SPGN file and enumerated graph variable (GV) sets. The parser creates a validated graph object, a data structure representing the input graph. Error checking eliminates any invalid SPGN. All values of variables affecting primitive execution are validated against constraints and requirements of the domain primitives. The graph object represents a flattened graph in which all subgraphs and family constructs have been expanded. GrTT's graph analysis object creates a state machine behavior specification from the graph object and behavior data provided by the domain primitive database. Any behavior error conditions are determined at this point. An example of such an error might be a graph with a periodic execution sequence that would be too long to code or would require too large a memory map. This long periodic execution sequence is normally caused by an ill-advised combination of node execution parameters. GrTT's autocoder object generates an Ada procedure implementing the state machine specification for all GV value sets. This Ada procedure becomes the primitive for an equivalent node replacing the partition in the original domain primitive application. A single equivalent node graph containing the procedure as its primitive is also generated by the autocoder object.

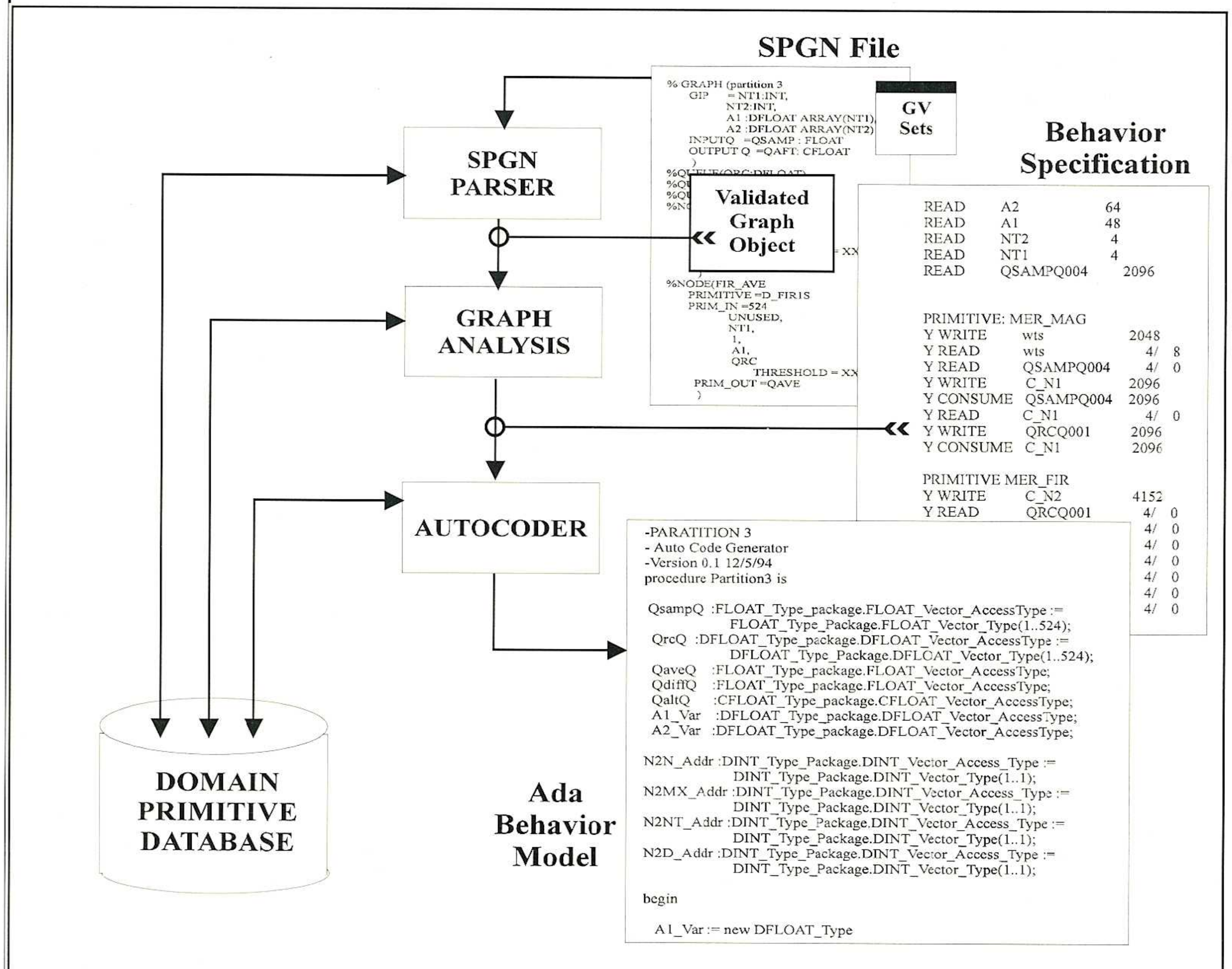


```
% GRAPH (partition 3
    GIP      = NT1:INT,
               NT2:INT,
               A1 :DFLOAT ARRAY(NT1),
               A2 :DFLOAT ARRAY(NT2)
    INPUTQ   =QSAMP : FLOAT
    OUTPUT Q  =QAFT: CFLO
    )
%QUEUE(QRC:DFLOAT)
%QUEUE(QAVE:FLOAT)
%QUEUE(QDIFF:FLOAT)
%NODE(MAG
      PRIMITIVE = D_MAG
      PRIM_IN =524
              QSAMP
                    THRESHOLD =
      PRIM_OUT = QRC
      )
%NODE(FIR_AVE
     PRIMITIVE =D_FIR1S
     PRIM_IN =524
              UNUSED,
              NT1,
               1,
               A1,
               QRC
                    THRESHOLD =
      PRIM_OUT =QAVE
      )
```

```
/* PARTITIONS -Auto Code Generator - Ver
main()
{
  int          *NT1;
  int          *NT2;
  double        *A1;
  double        *A2;
  float        * QSAMPQ004;
  COMPLEX_FLOAT    *QAFTQ005;

 double       dfA_N2(6);
 float        fA_N2(6);
 int          i_N2, j_N2, k_N2, M_N2, L_N2;
 double        dfA_N3(8);
 float        fA_N3(8);
 int          i_N3, j_N3, k_N3, M_N3, L_N3;
 int          i_N4, j_N4;
 float       two_N4;
 READ_VAR (A2, 8, 0, 6240);
 READ_VAR (A1, 6, 0, 6304);
 two_N4 = 2;
```

```
-PARATITION 3
- Auto Code Generator
-Version 0.1 12/5/94
procedure Partition3 is

 QsampQ  :FLOAT_Type_package.FLOAT_Vector_AccessType :=
            FLOAT_Type_Package.FLOAT_Vector_Type(1..524);
  QrcQ  :DFLOAT_Type_package.DFLOAT_Vector_AccessType :=
            DFLOAT_Type_Package.DFLOAT_Vector_Type(1..524);
  QaveQ   :FLOAT_Type_package.FLOAT_Vector_AccessType;
  QdiffQ   :FLOAT_Type_package.FLOAT_Vector_AccessType;
  QaltQ    :CFLOAT_Type_package.CFLOAT_Vector_AccessType;
  A1_Var   :DFLOAT_Type_package.DFLOAT_Vector_AccessType;
  A2_Var  :DFLOAT_Type_package.DFLOAT_Vector_AccessType;

N2N_Addr :DINT_Type_Package.DINT_Vector_Access_Type :=
            DINT_Type_Package.DINT_Vector_Type(1..1);
N2MX_Addr :DINT_Type_Package.DINT_Vector_Access_Type :=
            DINT_Type_Package.DINT_Vector_Type(1..1);
N2NT_Addr :DINT_Type_Package.DINT_Vector_Access_Type :=
            DINT_Type_Pack
N2D_Addr :DINT_Type_Packa
            DINT_Type_Pack

begin

  A1_Var := new DFLOAT_Typ
```

Figure 1. GrTT Behavior Modeling of PGM Graphs. GrTT produces an Ada behavior model of a DSP program translated from a PGM graphical specification.



```
% GRAPH (partition 3
    GIP     = NT1:INT,
              NT2:INT,
              A1 :DFLOAT ARRAY(NT1),
              A2 :DFLOAT ARRAY(NT2)
    INPUTQ  =QSAMP : FLOAT
    OUTPUT Q  =QAFT: CFLOAT
    )
%QUEUE(QRC:DFLOAT)
%QU
%QU
%NO
                                      = XX
    )
%NODE(FIR_AVE
    PRIMITIVE =D_FIR1S
    PRIM_IN =524
              UNUSED,
              NT1,
               1,
               A1,
               QRC
                  THRESHOLD = XX
     PRIM_OUT =QAVE
      )
```

```
READ        A2            64
READ        A1            48
READ        NT2            4
READ        NT1            4
READ        QSAMPQ004        2096


PRIMITIVE: MER_MAG
Y WRITE       wts            2048
Y READ        wts               4/   8
Y READ        QSAMPQ004         4/   0
Y WRITE       C_N1           2096
Y CONSUME     QSAMPQ004      2096
Y READ        C_N1              4/   0
Y WRITE       QRCQ001        2096
Y CONSUME     C_N1           2096

PRIMITIVE MER_FIR
Y WRITE       C_N2           4152
Y READ        QRCQ001           4/   0
                                4/   0
                                4/   0
                                4/   0
                                4/   0
                                4/   0
                                4/   0
```

```
-PARATITION 3
- Auto Code Generator
-Version 0.1 12/5/94
procedure Partition3 is

 QsampQ  :FLOAT_Type_package.FLOAT_Vector_AccessType :=
            FLOAT_Type_Package.FLOAT_Vector_Type(1..524);
  QrcQ  :DFLOAT_Type_package.DFLOAT_Vector_AccessType :=
            DFLOAT_Type_Package.DFLOAT_Vector_Type(1..524);
  QaveQ   :FLOAT_Type_package.FLOAT_Vector_AccessType;
  QdiffQ  :FLOAT_Type_package.FLOAT_Vector_AccessType;
  QaltQ   :CFLOAT_Type_package.CFLOAT_Vector_AccessType;
  A1_Var  :DFLOAT_Type_package.DFLOAT_Vector_AccessType;
  A2_Var  :DFLOAT_Type_package.DFLOAT_Vector_AccessType;

N2N_Addr :DINT_Type_Package.DINT_Vector_Access_Type :=
             DINT_Type_Package.DINT_Vector_Type(1..1);
N2MX_Addr :DINT_Type_Package.DINT_Vector_Access_Type :=
             DINT_Type_Package.DINT_Vector_Type(1..1);
N2NT_Addr :DINT_Type_Package.DINT_Vector_Access_Type :=
             DINT_Type_Package.DINT_Vector_Type(1..1);
N2D_Addr :DINT_Type_Package.DINT_Vector_Access_Type :=
             DINT_Type_Package.DINT_Vector_Type(1..1);

begin

  A1_Var := new DFLOAT_Type
```

Figure 2: Graph Translation tool Architecture and domain primitive Database Support. Ada behavior models are autocoded intermediate behavior specifications translated from PGM graphical specifications.

The behavior models generated by GrTT may be used to fulfill several important HW/SW codesign functions. GrTT software partition behavior models may be used to validate target-specific autocoded executables. The single node graph with a GrTT behavior model embedded as its primitive may be used to validate the partition translation and generate test vectors for other target- specific translations of the team's autocoding process. GrTT behavior models may be embedded as equivalent nodes' primitives in an equivalent graph generated during software architecture verification in the team's codesign process. Equivalent graph execution using GrTT behavior models will support validation of application requirements capture through the translation process. Since Ada syntax is used in VHDL, Ada procedures implementing behavior of PGM graph partitions will be common for hardware and software implementations. Because of this, GrTT behavior models of hardware partitions may be embedded as the procedural part of a VHDL behavior architecture, thus automating generation of VHDL behavior models from graphical architecture specifications. GrTT may be used to support hybrid, multi-level VHDL simulations. GrTT will produce a behavior system model from a graphic specification that may be used in a hybrid simulation with a lower level structural model of a system component. The GrTT behavior model will simulate all behavior of the system not included in the lower level model.

Figure 3 illustrates the execution of a GrTT generated behavior model of a PGM graph partition. Input and output vectors are shown above the single node graph with a GrTT generated behavior model primitive. The model may be generated with "taps" specified for any internal partition queues the user may wish to view. Behavior of those tapped queues will be generated during execution, thus providing the user with a virtual oscilloscope view of internal partition behavior. A demonstration version of GrTT will be completed by October, 1995. A beta version will be available for evaluation in the first quarter of the calendar year 1996. A commercial release, as part of MCCI's autocoding toolset, is planned for 1996.

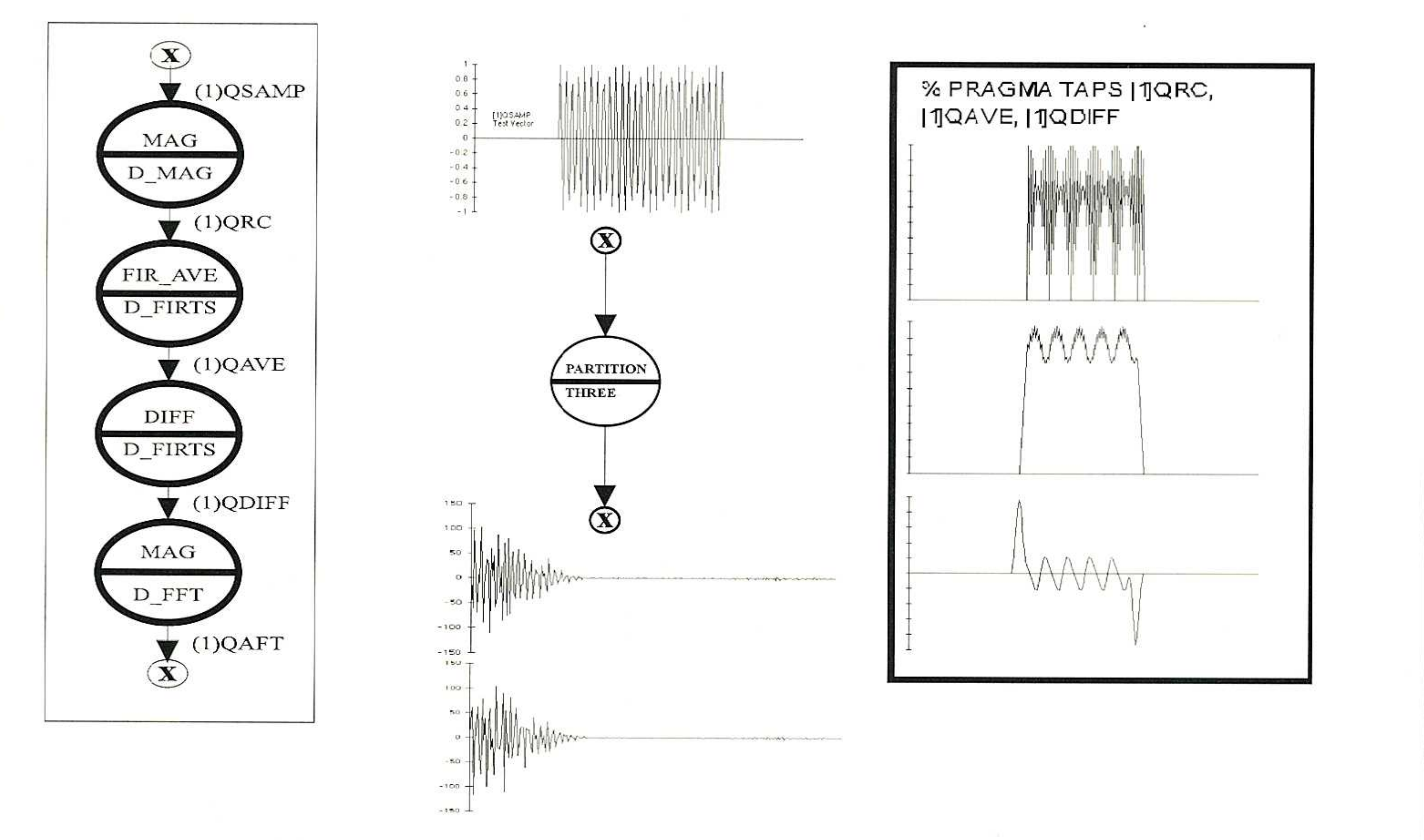


Figure 3. GrTT Behavior Model Executin Output.

RASSP Working Group Discusses Terms and Taxonomies

by Carl Hein, Leader, RASSP VHDL Modeling Terminology Workgroup

During the "Workshop on RASSP VHDL Modeling" at the January RASSP Principal Investigators Meeting in Atlanta, GA, RASSP contributors discussed how to adopt a consistent modeling terminology and taxonomy. A consistent terminology will facilitate communication among the RASSP participants by providing a common language where everyone knows and understands the terms. Currently, many participants use different words for similar concepts and do not correspond on the meaning of common terms.

At the PI meeting, a VHDL Modeling Terminology/Taxonomy working group composed of by Randy Harr, ARPA RASSP Program Manager, Todd Carpenter of Honeywell Technology Center, Carl Hein of Martin Marietta, Paul Kalutkiewicz of Lockheed Sanders, and Vijay Madisetti of Georgia Tech. was formed.

This article focuses on the working group's discussion of ideas presented by Martin Marietta, and the group's attempt to take a first step toward developing a universally acceptable set of terms and taxonomy.

1. Initial Terminology

The working group suggested that current modeling terms that are generally regarded as useful and well understood by the RASSP community (shown in Table I) become the base of a common RASSP terminology. The working group will then refine the definitions and names, and add other terms as needed for RASSP.

2. Initial Taxonomy

The terms in the table are grouped into several classes. System-level refers to models for which there is not a prior notion of hardware or software implementation details. Other terms refer to the hierarchy of detailed hardware and software or describe general model content, such as behavior, function, and structure. And there are terms that refer to parallel hierarchies, such as the structural hierarchy.

3. Comparing Previous and Proposed Taxonomies

The working group compared three model definition approaches (shown in Table II) to draw parallels and identify consistent threads.

The Ecker1 and Madisetti spaces have two axes in common; their remaining axes do not directly correspond. Both have an axis for "Time" resolution and a second axis that represents the resolution of "Values" in a model. Ecker calls the second axis "Value," while Madisetti calls it "Format." The Gajski-Kuhn Y- chart has a similar axis called "Functional-Representation." The third axis of the Ecker cube is similar to the "Structural-Representation" axis of the Gajski-Kuhn Y-chart, but has no corresponding axis in the Madisetti case.

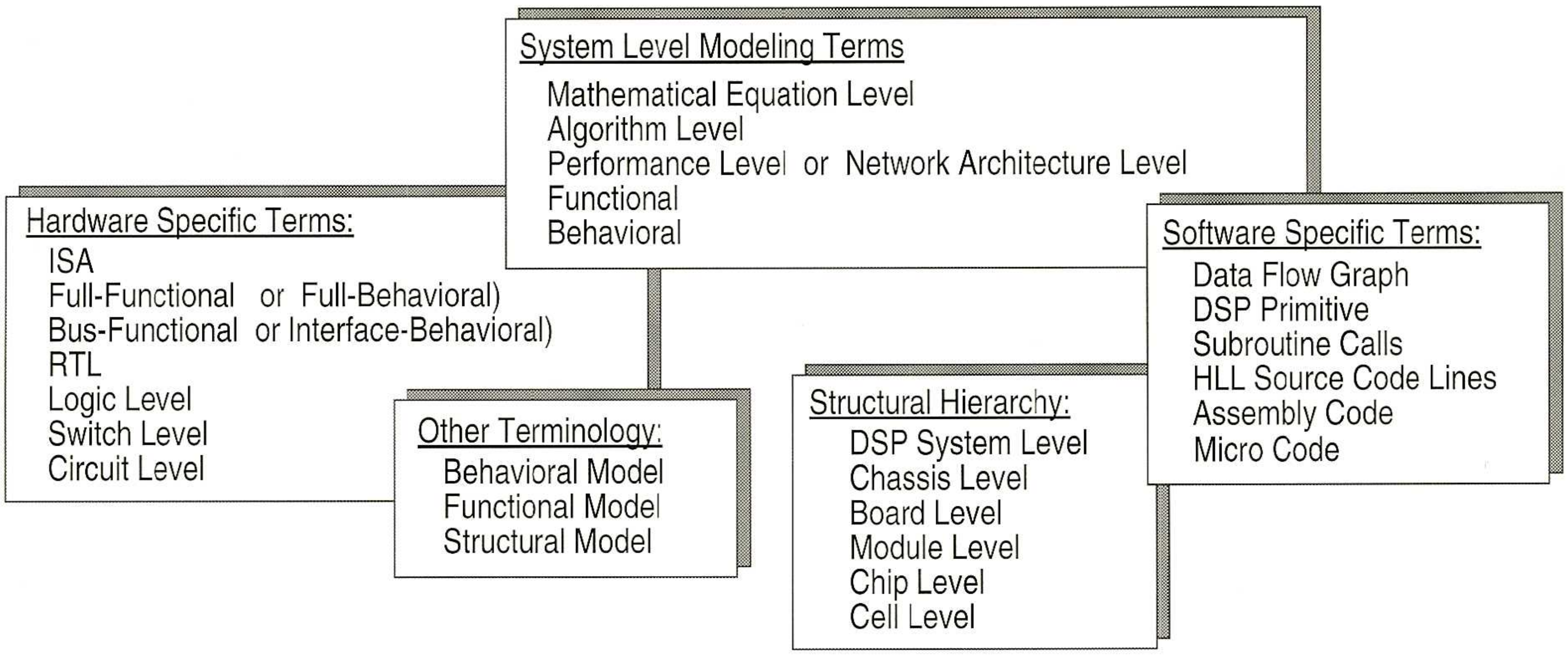
None of the remaining axes of the taxonomies directly correspond to each other. The Y-chart seems limited, and none of the methods appear to directly address the hardware/software codesign aspect.

4. New Axes

After examining the previous taxonomies, the working group discussed various types of axes that might more clearly represent model attributes relevant to a RASSP designer. Martin Marietta emphasized selecting relatively simple axis names and concepts that are quickly understood. This goal was to enable wide acceptance among new students and working design engineers in the industry.

First, a common set of attributes were proposed to describe a model's resolution, both internally and externally, but independently. Distinguishing between the two views is important in selecting, using, and building models because it enables clarity

Table I - Common Abstraction Level Terms



System Level Modeling Terms
- Mathematical Equation Level
- Algorithm Level
- Performance Level or Network Architecture Level
- Functional
- Behavioral

Hardware Specific Terms:
- ISA
- Full-Functional or Full-Behavioral)
- Bus-Functional or Interface-Behavioral)
- RTL
- Logic Level
- Switch Level
- Circuit Level

Other Terminology:
- Behavioral Model
- Functional Model
- Structural Model

Structural Hierarchy:
- DSP System Level
- Chassis Level
- Board Level
- Module Level
- Chip Level
- Cell Level

Software Specific Terms:
- Data Flow Graph
- DSP Primitive
- Subroutine Calls
- HLL Source Code Lines
- Assembly Code
- Micro Code

Table II - Prior work

Source - Taxonomy	Axes						
Gajski and Kuhn: Y-chart		Funct. Rep.		Struct. Repr	Geom. Rep		
Ecker: Ecker cube	Timing	Value	View				
Madisetti: RASSP Taxonomy	Timing	Format				Value	State

and precision. Existing terminology often mixes attributes, as viewed from inside a model, from similar attributes, as viewed from outside the model. The previous taxonomies had a common set of attributes that applies to both cases.

Next,Martin Marietta realized that what characterizes a model in every case is its relative "resolution of details" of some type or another. This means that the axes should all be in terms of resolution. The group identified four orthogonal aspects that are described in various degrees of resolution:

1) Timing detail
2) Value detail
3) Structural details
4) Functional details

The proposed taxonomy ended up with eight attributes to describe a model's level of description:

1. Internal Resolutions: timing, value, structure, and function.
2. External Resolutions: timing, value, structure, and function.

Note that this attribute set does not describe how a hardware model appears to software. The eight attributes do not address the hardware/software codesign aspect. To remedy this, the group proposed a ninth axis to represent the level of software programmability of a hardware model or, conversely, the abstraction level of a software component in terms of its complementary hardware model where it executes. Figure 1 shows the axies with example resolutions. The text box explains the corresponding axes in greater detail.

The working group discussed several examples of mapping common model terms onto the axes. Each model type was easily distinguished from the others on the new axes. Though outside the scope of the meeting, working group participants discussed positioning model types on the axis. The next step in the consensus-building process is the concise definition of the modeling terms relative to the axes.

5. For More Information

RASSP participants are encouraged to consider and comment on the terms and taxonomies presented here, and to offer terms that were overlooked. Although not completed, the group's intention is to assign concise definitions to the listed terms. For details, please contact Carl Hein by phone at 609-866-6541 or by email at chein@atl.ge.com.

References

[1] Ecker, W. and Hofmeister, M., "The Design Cube - A Model for VHDL Design Flow Representation," Proceedings of the Euro-VHDL, 1992, pp. 752-757.

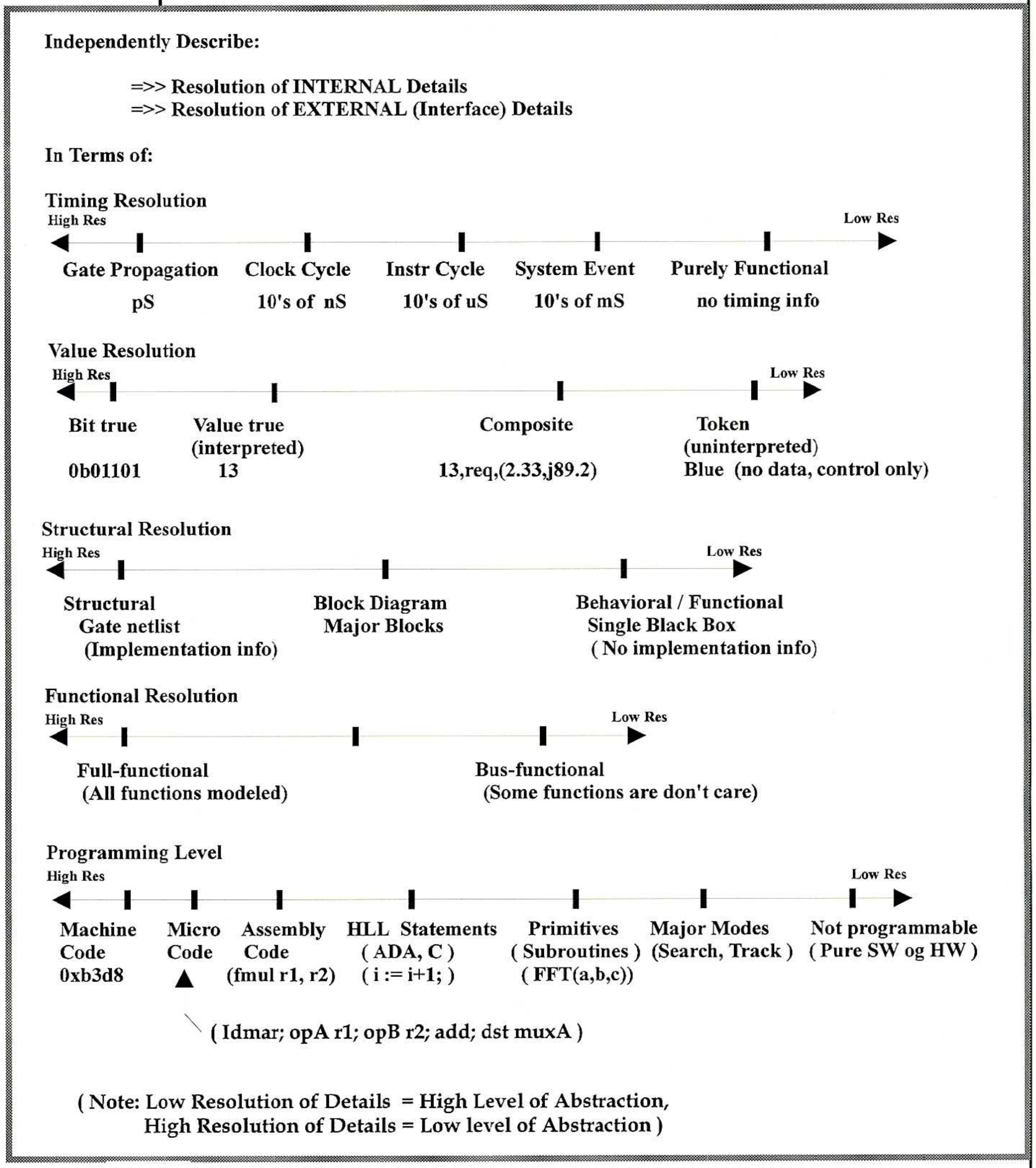


Figure 1. Proposed Taxonomy

Further Explanation of Axes presented in Figure 1.

Time Resolution

The Time Resolution axis represents the resolution of events that are modeled in terms of their time scale. Resolution is analogous to precision, which is to be distinguished from accuracy. For instance, a model's time resolution may be stated in terms of the starting and ending times of major system functions, where each function spans thousands of clock-cycles. In such a case, we say the model resolves events down to the major function level and not down to the clock-cycle level, even though the accuracy of the starting and stopping times may be specified accurately to within one clock-cycle.

Value Resolution

The Value Resolution axis represents the resolution with which values are specified in a model. Again, note that resolution is analogous to precision, as distinguished from accuracy. For instance, both 3.09 and 3.10 contain three digits of precision, yet the later represents the value 3.1000 more accurately. Similarly, a 3-bit hardware register containing the value negative-one (-1) may be modeled with high resolution in terms of its actual two's-complement binary "0b111" (or signed-magnitude binary "0b101") form, or it may be modeled more abstractly as a signed integer "-1" or even floating-point value -1.0E+00. All are equally accurate, but the first instance most precisely resolves the value to its form as actually contained in the target device. The more abstract the representation of a value is, the lower its resolution of implementation details for representing it.

Structural Resolution

The Structural Resolution refers to the level of information detail a model provides about how the modeled component is constructed out of constituent parts. For example, one model of a processor chip may have no information about its internal structure. A second model of the same chip may specify its structure in terms of five major blocks. A most detailed model might specify the internal structure in terms of the interconnection of specific logic gates.

Although more abstract, the second model is perfectly accurate as long as the five major blocks can be identified as connected in the gate-level model. This understanding of structural resolution holds for both external structure and internal structure, as described in this example. For instance, a port on an abstract model may be a single composite value with many fields, but no information about the physical structure. A high-resolution model would specify the ports' structure in terms of bit-widths, address and data bus, and hand-shaking lines.

Functional Resolution

The Functional Resolution axis refers to the level of detail with which the functionality of a component or system is modeled. For instance, a highly abstract model might specify the function of a digital filter in terms of its mathematical transformation, while a high-resolution model might resolve the function in terms of the boolean operations that implement the target device. Both models can be functionally accurate.

In the extreme, the most abstract (or low-resolution) model might contain no functionality at all. As with internal functionality, the external functionality specifies the interface behavior of a device's (or system's) ports.

Software Programming Resolution

The Software Programming Resolution axis refers to the level of granularity with which a hardware component may be programmed. More accurately, it is the level of instructions that the model of a hardware component interprets in executing target software.

For instance, a network performance model only interprets instructions on the level of DFG primitives, such as MATINV, VMUL, or FFT. Such primitives often represent hundreds of lines of source code, but are interpreted as a single instruction in terms of a time-delay by a network performance model. An ISA model interprets individual assembly instructions. In this sense, the ISA model is programmable at a much finer granularity, or higher resolution, than the network performance model.

At the lower extreme, a model of a micro-code programmable processor is programmable at an even lower level of granularity than the ISA model, since it allows control of individual register and multiplexer structures within the device during execution of an assembly-level instruction. At the opposite extreme are software design components or non-programmable models, since neither in itself interprets programmable instructions.

RASSP Digest-Rapid Prototyping of Application Specific Signal Processors

The RASSP Digest is published quarterly and provides information for and about the RASSP Program and rapid systems development. For more information, contact Dr. Anthony Gadient or Dr. Vijay Madisetti, Editors, at the addresses below:

Dr. Anthony J. Gadient
Phone: 803-760-4082
FAX: 803-760-3349
Email: gadient@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

Dr. Vijay K. Madisetti
Phone: 404-853-9830
FAX: 404-853-9171
Email: vkm@ee.gatech.edu
Georgia Tech
Sch. of Elec. & Computer Eng.
Atlanta, GA 30332-0250

Kristi Adams
Managing Editor
Phone 803-760-3376
Email: adamsk@scra.org

RASSP Steering Committee

ARPA (ESTO)	
-Randy Harr	**Program Manager**
ARMY	
-Randy Reitmeyer	**Administrative COTR, Martin Marietta**
-Arne Bard	**Technical COTR, Martin Marietta**
NAVY	
-Ingham Mack (ONR)	
-Gerry Borsuk (ONR)	
-Joe Killiany (NRL)	**Administrative COTR, Lockheed/Sanders**
-J. P. Letellier (NRL)	**Technical COTR, Lockheed/Sanders**
AIR FORCE	
-Stan Wagner	**Educator Facilitator and Technology Base**
-John Hines	**COTRs**

Announcing Summer '95 RASSP Short Courses

The RASSP Education & Facilitation team will offer two short courses on key advances of the RASSP program. Each of these courses is intended for design engineers and will provide extensive hands-on exercises.

Rapid Prototyping Using VHDL Topics Include:

- Basic/Structural VHDL
- Behavioral VHDL- RASSP Methodology Overview
- Virtual Prototyping Using VHDL
- System Level Modeling
- System Level VHDL
- HW/SW Codesign
- Hardware Synthesis Overview
- Test Technology Overview
- Libraries : Generation, Maintenance, and Documentation

Led by: Prof. J. Aylor, University of Virginia
Course Dates: August 7-11, 1995
Location: Boston, MA

Algorithm and Architectural Design and Prototyping of Embedded DSP Systems Topics Include:

- RASSP Methodology Overview
- Algorithm/Functional Design for Signal Processors
- DSP Architectures
- Scheduling and Assignment for DSP
- Virtual Prototyping for DSP Architectures
- HW/SW Partitioning
- Communication and I/O Architectures

Led by: Prof. V. Madisetti, Georgia Institute of Technology
Course Dates: August 22-25, 1995
Location: Phoenix, AZ

Enrollment will be limited to 20 attendees per course. A registration fee of $350 will be charged to cover lunches, refreshments, course material, and other administrative expenses. To Register for the short courses you can phone RASSP E&F at 803-760-3376, E-mail us at courses@rassp.scra.org or register via the World Wide Web at http://rassp.scra.org/courses.html

Calendar of Events

2nd Annual RASSP Conference For More Information: Mark Feuchter 703-351-8463	July 24-27, 1995	Arlington, VA
RASSP Short Course Rapid Prototyping Using VHDL For More Information: 803-760-3376 WWW http://rassp.scra.org	August 7-11, 1995	Boston, MA
RASSP Short Course Algorithm and Architectural Design and Prototyping of Embedded DSP Systems For More Information: 803-760-3376 WWW http://rassp.scra.org	August 22-25, 1995	Phoenix, AZ
VIUF Conference For More Information: Pam Rissman 415-329-0578 or fax: 415-324-3150	October 15-18, 1995	Boston, MA
CALS EXPO For More Information: Dr. D. Brent Pope 202-775-1440 or brentpope@delphi.com	October 23-26, 1995	Long Beach, CA
DSP World Expo For More Information: Ann Harris 617-891-6000 or DSPWorld@world.std.com	October 24-26, 1995	Boston, MA

SCRA
5300 International Blvd.
N. Charleston, SC 29418

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • MMG • ADL

RASSP Digest

RASSP - Rapid Prototyping of Application Specific Signal Processors

Vol. 2, 3rd. Qtr. 1995

RASSP at 24 Months

Methodology

RASSP
Reinventing
Electronic
Design

Architecture
Infrastructure

ARPA • Tri-Service

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • MMG • ADL

In This Issue

RASSP at 24 Months

Vijay K. Madisetti & Anthony J. Gadient

We welcome the readers to this special issue of *The RASSP Digest* that is devoted entirely to the achievements and accomplishments of the RASSP primes as documented in the Second Annual RASSP Conference, held between July 24-27, 1995 in Crystal City, Virginia. RASSP is now 24 months old with a wide variety of accomplishments to date. The demonstrations and presentations at the RASSP conference have clearly shown that 4X is within reach, commercialization of RASSP technological breakthroughs is occuring, and that users, vendors, and suppliers are accepting executable specifications, virtual prototyping, VHDL-based system design automation, reuse libraries, enterprise integration and workflow systems, and virtual corporations as ideas that are no longer constrained to the drawing board, but whose day has come.

As RASSP enters its third year, the RASSP Educator and Facilitator (RASSP E&F) program has offered short courses and numerous management tutorials to a variety of industrial organizations. As the transition phase of the RASSP E&F program ramps up, the RASSP E&F team will help to ensure the successful transfer of the RASSP technology by continuing to facilitate the successful progression of organizations through a four phased technology transfer process of (1) developing awareness of the RASSP technology, (2) generating understanding of the benefits of the RASSP technology, (3) working with the RASSP primes to support the use of RASSP technology on selected pilot programs, and (4) obtaining organizational commitment to RASSP by incorporating the RASSP methodology and technology into daily business practice as the ultimate goal.

This special issue of the *Digest* indicates that the time is right for phase (3). As the old adage goes, the "proof of the pudding is in the eating." *Bon appetit*, for there is a lot to *Digest* in this issue.

Vijay K. Madisetti
School of Electrical
& Computer Engineering
Atlanta, GA 30332-0250
vkm@ee.gatech.edu

Dr. Anthony J. Gadient
SCRA
5300 International Blvd.
N. Charleston, SC 29418
gadient@scra.org

The Second Annual RASSP Conference -- A Mid-Program Review

Randy Harr

The second annual RASSP conference, recently held in Crystal City, Virginia from July 24-27, 1995, served as a turning point for the RASSP program in many ways. First, now that the program has been underway for two years, there were significant results to show. Second, it was the first real chance the community at large has had to look at all facets of the RASSP program in one location and see the breadth and depth it has. Last, it was the halfway point of the program and served as a marker to look back at where we have come from and begin to focus on the realistic results of the program.

The conference consisted of three major activities: a detailed exhibition and demonstration hall, a concurrent technical program, and the tutorial program. Additionally, side "birds-of-a-feather" meetings and ad-hoc interactions served to focus many researchers in the community to discuss the problems they overcame and the new ones they face.

The expanded exhibition was a highlight for this year's conference as over 300 attendees saw demonstrated the latest advances in tools for designing large DSP systems and the results of applying the tools to some real-world design problems. A theme throughout many of the booths was the results of the first benchmark -- the architecture design and virtual prototype of the SAR image formation processor. From the executable specification to the detailed virtual prototypes, people were able to follow the design process of a high performance, parallel DSP implementation. Also highlighted were the many booths from the RASSP technology base development and the corollary, non-funded, commercial market for DSP systems. Overall, the exchange was very beneficial for the RASSP and DSP system community.

The technical program concurrent with the exhibition was focused around ten major themes. These themes were the introduction to RASSP and its second year accomplishment, demonstrations of the RASSP process, projecting RASSP benefits, systems performance modeling, HW/SW development processes, VHDL prototyping, benchmark results, model year architectures, design process management and novel design approaches. Each provided in depth coverage of development details of the front to back end design tools and processes.

In addition to the technical program, a tutorial day focused on providing in depth information about specific, important topics. The topics were Ptolemy, RASSP Design for test (DFT) methodology, and VHDL-based, top-down virtual prototyping for large DSP systems. Ptolemy is a significant new develop-ent in the RASSP program under the guidance of Dr. Edward Lee of the University of California, Berkeley. The results to date of this co-funded effort (the industry funds the other half) have already had a wide ranging impact with major EDA vendors, commercial mixed-signal developers, and the research community. The tutorials and the conference as a whole represent a significant effort within the RASSP program to focus on adoption and proliferation during the program development.

The first day of the general sessions and exhibits was also focused into an overview day which was separately promoted and drew an extra 75 attendees. It started with a keynote address by Dr. Robert Kahn, President, CNRI, and one of the founding developers (both in industry and at ARPA) of the ARPAnet and related technologies. His informative talk on the infrastructure needs of the National Information Infrastructure provided some insight into the methods being taken to re-build the base of the quickly expanding Internet so that virtual enterprises, large scale design, and commerce can occur over this shared and open resource. Following was an overview of the RASSP program which highlighted the early insertions and success of applying the RASSP technology to real system design problems.

Specifically, RASSP over the past year has been successful in the design of a back-end IRST companion to the ARPA sponsored Airborne InfraRed Measurement System (AIRMS) -- a high resolution IR detector and test station. By simply applying VHDL-based virtual prototyping to this project they realized a 2.2x overall savings in design time -- mostly coming from a very short HW/SW integration and test activity. This demonstration (a model year 0 excercise in IRST development of which you will see more in the coming years) was conducted over a virtual enterprise consisting of Hughes Aircraft (CA), Motorola (AZ) and Sanders, a Lockheed Martin Company (NH).

Additionally, a prevalent highlight was the UAV SAR image formation benchmark in which MIT Lincoln Labs developed an executable specification (in VHDL) and a real-time data source/sink. Two contractors then went off to design and build solutions -- first virtually and then in real hardware and software. The virtual prototype was used to assess performance, verify design criteria and validate critical software pieces before actually committing to hardware or full software development. A third contractor (Mitre) simply took the software specification and ported it to an Intel Paragon multi-processor computer to assess the viability of using general purpose, high performance computing architectures for this type of DSP algorithm. You are encouraged to get a copy of the proceedings and learn more about this exciting project.

The RASSP program, for the first time, is putting VHDL to use at all layers of abstraction in the design process. The significance is that this is the first, real documented case of using VHDL from executable specifications through performance analysis, system level modeling and down to RTL/Gate level synthesis -- a feature that has been touted as a capability for many years. A significant result of the RASSP program is the putting into practice the process and tools to support using a single language for [digital] functional specification from the requirements down to the final gates of the hardware. It is requiring real issues to be addressed such as how to deal with algorithm and software development in a language based specification refinement environment and how to define layers or stages in the design process at which models or detail should be defined.

Much is going on in the RASSP program and will continue to become apparent over the coming year. New, linked architecture analysis and design tool suites, taxonomies for modeling in VHDL at many abstraction levels, process modeling/ tracking/metric tools and support, model year architectures for simpler upgrades, design for test, links to manufacturing, more professional development and university courses, and many others. Please make sure to attend our third conference next July to see the developments and provide additional insight to the team to assist them in solving your problems.

Randy Harr
ARPA
370 North Fairfax Road
Arlington, VA 22203-1714
rharr@arpa.mil

The Second Annual RASSP Conference, Synopsis of Session 2, "Introduction to RASSP and 2nd Year Overview"

Mark Richards

The two RASSP prime contractors, Lockheed-Martin Advanced Technology Laboratories (ATL) and Lockheed-Martin Sanders, have the daunting job of developing a well-founded vision of the methodology needed to achieve real improvments in the DSP prototyping process, creating and assembling the technologies needed to implement the vision, and demonstrating that it works in applications of real importance to the Department of Defense. The intent of this first technical session of the conference, titled "Introduction to RASSP and 2nd Year Overview," was to provide the attendees with a mid-term snapshot of the two development efforts: their vision, the new capabilities developed and demonstrated to date, and the steps still needed to realize the RASSP 4X goals.

Bill Hood, Sanders RASSP Program Manager, presented the overview of the Sanders effort. Mr. Hood emphasized the critical role of top-down design and VHDL to build virtual prototypes of new DSPs as the key methodology concept, attributing a 2.2 speedup in design time to these techniques in a flight hardware test case. He went on to describe Sanders' ENTIRE design environment and tools suite, with special attention to the role of data management tools and to the Sanders experience to date in establishing and working in a "virtual corporation" environment.

The ATL effort was summarized by its program manager, Jim Saultz. The ATL development approach has emphasized enhancing and integrating best-of-class technologies required to realize a RASSP design environment. This approach has already benefited the general digital design market through a number of new commercial tool offerings from new and established EDA companies. Mr. Saultz described ATL's efforts in defining an effective approach to DSP architecture that would support the RASSP goals of hardware/software codesign and model year product upgrades and then mapping this approach to EDA tools. Like Sanders, ATL stressed the importance of data management in building a complex RASSP design system.

Mark Richards
GA Tech
SEAL/RSD CCRF
Atlanta, GA 30332-0800
mark.richards@gtri.gatech.edu

Sanders RASSP Program Overview

W. Hood, M. Hoffman, J. Malley, C. Myers, R.Ong, E. Rundquist, L. Scanlan, F. Shirley, D. Uyemura

Abstract

This Sanders-led team of Motorola, Hughes and ISX has met all of the primary RASSP program objectives during the first two years of the program. This paper reviews the goals of the program and the unique ways in which our team is meeting them. The flexible methodology and design environment are described along with the progress made in creating a standard enterprise framework. The progress of the demonstration and benchmarking effort is detailed, as is the work towards proliferation of the RASSP process. The emphasis on VHDL and Ada-based virtual prototyping and its impact on Model Year Upgrades is discussed. The creation of the Virtual RASSP Corporation and the special Internet communication protocols developed to support the program are reviewed. Accomplishments in each of the program areas are reviewed along with specific goals for the next year of the program. Particular emphasis is placed on our Model Year 0 demonstration in which we designed, fabricated, and tested Infrared Search and Track (IRST) flight hardware in less than a year. Comparison of the time and resources required to perform Model Year 0 with a comparable non-RASSP development demonstrates that we have already achieved a factor of more than 2.2 X improvement in development time and development cost.

1. Introduction - RASSP Works!

RASSP is a Weapon System Development Process

RASSP is a DoD program to develop a process that meets four ambitious goals. Over the four-year life of this ARPA project the cost and development time of upgrading and replacing embedded digital signal processors is to be reduced by a factor of four (4 X). At the same time the program is to provide for the marked lowering of life cycle costs and the development of weapon systems that work correctly the first time. At this halfway point of the program it is appropriate to measure progress toward these goals.

1.1 How Will We Reach 4 X and How Are We doing So Far?

We have chosen four major approaches which, when successful, will result in exceeding the goals of the RASSP program. In the order of most contribution to least, these are the goals:

- **Use "Top-Down" Design and VHDL Exclusively --** By enforcing the use of a **top-down design process** and using **Ada** and VHDL for all design work, we are able to build full-fidelity **Virtual Prototypes** for weapon systems processors. We have **designed, built, integrated, tested and delivered 3-D IRST flight hardware in 11 months at a cost of $3.5 Million. This is an improvement of more than 2.2 X in speed and cost over traditional "waterfall"** process methods.

- **Emphasize Reuse --** The intense cultivation of reuse as a philosophy, particularly with respect to software modules in Ada and hardware models in VHDL, has already begun to show reductions in design time, along with concurrent reductions in cost. **Reuse is supported by our RASSP Design Environment (RDE)** which includes new tools especially developed to promote reuse. **Reuse will contribute about 1.6 X to the 4 X goal.**

- **Produce an Integrated Design Environment -- ENTIRE (Environment and Tools for and Integrated RASSP Design Environment)** is demonstrating today a **breakthrough in tool integration.** Instead of the hundreds of encapsulations or integrations normally entailed in tying dozens of EDA tools together, ENTIRE needs only one per tool. **ENTIRE encourages reuse** by providing easier access to the many component libraries available today without necessarily having to use the specific tools normally associated with them. ENTIRE also enables our team to perform **distributed work using geographically separated Integrated Product Teams operating as Virtual Companies. We are on track to demonstrate at least a 1.5 X improvement with ENTIRE.**

- **Improve the Process -- The appropriate addition and extension of system tools in the ENTIRE suite** will allow industry-standard tools such as PGM, MATLAB, and Ptolemy to be easily used with each other and with other EDA tools. This helps ensure that the weapon systems that are built using RASSP **meet the users' requirements, work right the first time, and are in turn easy to upgrade.** These additions are necessary to meet RASSP first-pass quality goals, although generally we do not consider improvement of individual tools as part of our RASSP process.

1.2 What Weapon Systems Are We Using RASSP On Now?

New Hardware Development Programs Started

In the first 18 months of the program, we have started (and in some cases completed) six major processor development programs. These range from full-custom DSP hardware implementations to custom Ada software upgrades on COTS general purpose processors. We have also installed our RASSP ENTIRE environment at five sites outside the program, and are supporting these organizations as they learn to use it. Three other sites are scheduled to receive RASSP ENTIRE by January 1996.

- **Flight Hardware in 11 Months --** During the last year, we designed, built, and installed a processor system on an ARPA aircraft. This system is an upgrade to the signal processor in the ARPA Advanced Infrared Measurement System (AIRMS). **This development took less than 11 months and cost less than $3.5 million (Figure 1).** Based on independent measures this is 2.2 - 2.7 times faster and is about 1/3 the cost of previous industry standard effort. After building the "Virtual Prototype" in software, the

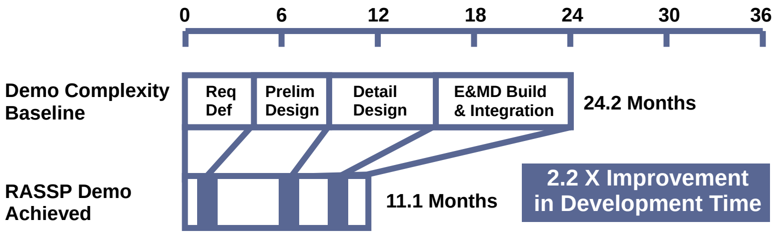


FIGURE 1. RASSP Demonstration of IRST hardware for AIRMS aircraft took 11 months, and improvement over standard practice of a factor of 2.2 in development time.

hardware was built one time and checked out and integrated in 6 weeks. Total board checkout time was less than two weeks of these 6 weeks.

- **F-14D IRST Hardware Upgrade --** Because of the reusable way (we call it “state-of-the-shelf”) in which RASSP approaches upgrades, we will be able to use much of the AIRMS work to replace seven processor boards in the F-14D Infrared Search and Track (IRST) weapon system with 2 identical boards. We will demonstrate this reduction in board count and complexity with flight hardware in 9 months, beginning this month. The current functionality of the IRST will be maintained.

- **F-14D 3-D IRST Systems Engineering Started --** Beginning in March 1996, we will apply the RASSP process to upgrading an F-14D IRST to include the most advanced of the new US Navy algorithms.

- **F-15 APG-63U Radar Upgrade USN F-18 APG-70 Radar Upgrade may Follow, Savings could Exceed $300 Million --** RASSP process and technology will be used to upgrade the F-15 radar signal processor. This program which starts in July and has the endorsement of the F-15 SPO, will upgrade the APG-63U. The Navy F-18 radar program "piggy-backs" an upgrade to the APG-70. This processor upgrade will take 18 months and will result in a a potential reduction in life-cycle costs of more than $300 million. Flight test is scheduled in early 1997 before the end of the RASSP contract.

- **US Navy/ARPA Fund ACOMMS Program --** We have successfully concluded a RASSP-based signal processor design contract for the US Navy. This acoustic program is called ACOMMS. Synthetic Aperture Radar Image Processor Built, Multiple Applications Found -- during the past year RASSP developed improvements to a Synthetic Aperture Radar (SAR) image processor designed as a payload for an unmanned air vehicle. This is an upgrade of a radar designed by Lincoln Laboratory. We are half way through this development which will produce working hardware in 12 months. Two classified applications of the design are being pursued.

- **JAST Uses RASSP --** Sander’s Advanced Engineering and Technology Division has won the JAST SAVE contract which includes funding for installation and support of the Sanders RASSP design environment in Fort Worth and Georgia for use in the development of the aircraft systems on the JAST program.

- **ENTIRE Finds New Uses and Users --** Our design environment has been installed at Woods Hole Oceanographic Institute, Sandia National Laboratory, and the Johnson Space Flight Center. It is currently scheduled to be installed at Lincoln Laboratory and the Wright Aeronautical Laboratories in 1995. Beta sites of our process are now being offered to the US defense industry without restriction or limitation.

2. Approach

In order to meet the goals of the program, a well-defined approach to implementing RASSP is required. This approach is based on four equally important program pieces. The first of these is to provide a **flexible methodology and environment** for users of the process. This process can then be easily adopted by the DoD industrial community. This allows a user to keep operating with those tools with which he is familiar. The second part of our approach is based on the provision to the user of a development paradigm which supports **large distributed environments** which are spread across **geographically separated** teams. This part of our approach includes tools for remote, cooperative work. Also included is a flexible environment which includes the incorporation of a **pay-per-use concept** for access to tools.

One of the central functions of our RASSP approach is the **RASSP Design Environment (Figure 2).** The RDE along with the EDA tools for a specific application becomes **ENTIRE (Environment and Tools for an Integrated RDE)**. Our development of the RDE and ENTIRE is aimed at solving the N-squared problem normally found in either encapsulating or integrating large numbers of tools to each other. Along with this goal go two more that are equally important. Finding a way that allows easier access to the huge number of commercially available data bases (component as well as VHDL and Ada), and controlling the configuration of programs developed in a distributed environment are major issues that our program is addressing.

We are building complete versions of the RDE and ENTIRE every four months. Each of these "builds" is evaluated and used by the rest of the program to do real work. Each year we formally release the RDE for external use. Concurrently with the build/release process we begin the development of the next release. This lets all of our team, our users and our customers see where we expect to be in the next year.

The release being used at this conference is the fourth build of the RDE. The RDE development effort uses previously developed RDE utilities in conjunction with the RASSP process of software development. This release of the RDE contains a common set of infrastructure services for use in a wide range of applications: **common desktop, automatic metrics collection, metrics analysis, reuse utility, technical review utility, problem reporting utility, log utility, and remote data access.**

The RDE is being tailored for use on our Model Year 1 demonstration to include domain specific tools, translators, libraries, and process support. The fully populated and tailored design environment is described as the ENTIRE concept. Our plans and progress on the RDE are reported in detail in “ENTIRE,” a paper by Ong, Costantino, and Philips presented at this conference.

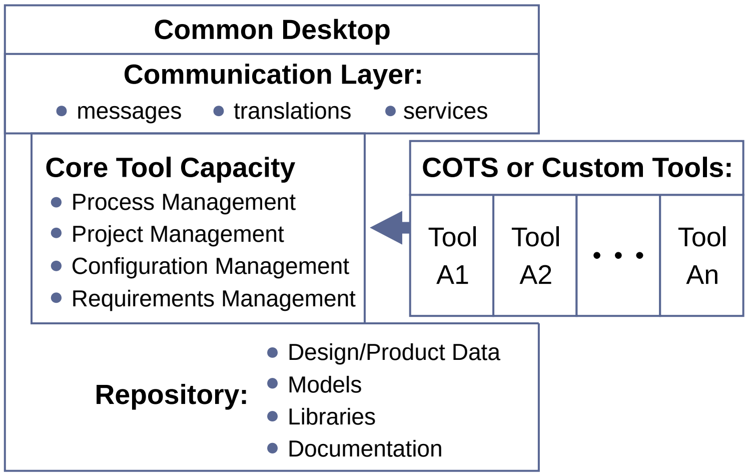


FIGURE 2. The RASSP Design Environment allows the insertion of either COTS or custom tools into an open framework which supports communications and design management.

The Sanders team has embarked on an ambitious Demonstration plan to execute three model year builds of flight hardware using our RASSP process. These Demonstration Model Years will add functionality to **two Infrared Search and Track (IRST) signal processing systems.** The first of these Model Years is complete and has resulted in **first-pass AIRMS flight hardware in 11 months** and a cost of $3.5 Million. The next model year will result in replacing seven different boards from WRA-2 in the **F-14D IRST** processor with 2 identical boards while maintaining the current 2-D functionality. The third model year will use the 5 empty slots to add **full 3-D capability** to the F-14.

In addition to the Demonstration, Sanders is executing a series of **Benchmarks** which measure the process improvement of our approach in an incremental way that makes full use of metrics as a gauge of progress. The first of these benchmarks involves the **virtual prototyping of a Synthetic Aperture Radar for application to an Unmanned Air Vehicle.** The second set uses RASSP to **upgrade the signal processing in the F-15 and F-18 radars.**

The fourth part of our approach is to **proliferate the process** widely. **Sanders is** dedicated to the proliferation of RASSP. An example is that the original concept of a facilitator on the program came from **Sanders**. Proliferation to a range of users provides a way of verifying our process. **Beta sites** also give us a way to get feedback on the strengths and weaknesses of our process.

2.1 Program Organization

In order to meet the goals of RASSP and to execute the approach that we selected, **Sanders teamed with Motorola, Hughes Aerospace, and ISX.** The technical work on the program is split nearly equally between Sanders, Motorola and Hughes while ISX has a small but significant role. These four companies have widely varying styles and business cultures. They are also spread out all over the United States. As a result, we have been forced to confront the issue of creating first a large virtual program and finally to face the challenge of creating a large virtual corporation.

Because of the success which Sanders and the US Air Force have had on the F-22 program with the use of Integrated Product Teams, we chose to use a similar management technique on RASSP. Our program consists of four **Integrated Process and Product Development Teams (IPPDTs)**. These four teams include **Systems, Design Environment, Demonstration, and Proliferation.** Sanders leads the Systems team as well as being the prime contractor on the program. Motorola has the lead responsibility on the Design Environment, while Hughes leads the Demonstration team. ISX has the responsibility for the Proliferation effort. All team companies share each of the team’s tasks. This leads to the concept of **“single responsibility, shared execution.”** This has worked well on the RASSP program.

Typical of the innovative ways used to **erase the geographical distances** separating the team members is the extensive use of the Internet. By using T-1 lines throughout the team, we are able, for example, to carry on **video conferences between desks** separated by thousands of miles. At the same time engineers can **share workstation screens, send encrypted files** between **fire-wall protected servers** around the country, and can access our MOSAIC server with its homepage directions to current RASSP activities. Concurrently, tools sited at one team’s location can be used remotely, or can be transferred **through various ftp protocols.** The Sanders team has set up a tiered ftp site for document access. Various levels of security and access are available, ranging from “Sanders only” through “Sanders plus Team” to “Team plus ARPA/Triservice Steering committee" to "Unrestricted.” This ftp site uses a document data base system that allows the customer access to more than 5000 documents created in the first half of the program, including all deliverables. This allows timely and cost effective review and commenting by the customer. It lowers delivery costs to zero for these reviews, and allows "final" delivery to take place instantaneously. Having the USG on-board as part of the team (including DPRO and USN contract monitors) opens the program, prevents surprises, and reduces costs.

2.2 RASSP Methodology

The Sanders methodology is definable by looking at its four fundamental parts. The first of these parts is a systematic and codified top-down design process integrated into an iterative approach to **model year development**. A second key attribute of the methodology is the emphasis on completing the hardware/software trade-off analysis before the system architecture selection.

The last two parts of the methodology are closely linked. We use a **virtual prototyping** technique in which a complete VHDL model is developed to reduce integration risks. The last piece of our methodology involves delivery of a complete description of the system as a VHDL model. This includes source code for all programmable processors in the design. All source code is written in Ada.

The emphasis on virtual prototyping improves the quality of design, documentation, and error checking. This reduces mistakes that can have costly consequences which do not appear until late in the development cycle. Our objective in RASSP is to produce a **top-down design methodology** in which a design is successively refined as a growing, verifiably consistent data package over the course of its development. The initial functional requirements are captured and ported into a simulation environment **supported by a VHDL simulator.**

This **"executable requirement"** is refined until it becomes a complete **"executable specification."** This acts as a test bench and also provides the requirements for the next lower level of the design. This top-down development process is used over the life cycle of a signal processor in an iterative way. These iterations allow for continuous improvement, i.e., RASSP Model Year Upgrades, of the signal processing system throughout its development and deployment. Upgrades can include functional improvements, repackaging for reduced power, cost, weight, or volume, or to replace parts out of production with those more readily available. The functional and performance specifications from one model year become the executable specification for the next. The hierarchical VHDL description provides for easy reuse of any downward chain, and redevelopment starting from any upward intact chain. The corresponding **Ada** code provides for software reuse in a similar manner.

2.3 RASSP Design Environment and ENTIRE

The RASSP Design Environment (RDE) from Sanders, Hughes Aircraft, Motorola, and ISX Corporation facilitates Integrated Product Development (IPD) by providing a collaborative work environment. The RDE provides support for automating the product development process. This will enhance the DSP product with respect to a 4X (four times) improvement in development time, cost, and quality. The **RDE enables the IPD philosophy** with its support of rapid iterations, incremental promotion, and scalable configuration management controls. The IPD approach can be employed during all phases of a product's life cycle from conceptual and detailed design through production to field support. A fundamental thread in supporting IPD is communication between individuals within and across teams of people, especially when the teams are not co-located. Therefore, it is of utmost importance that effective, secure, and timely communication of status, schedules, product data, and other information items be provided to all team members wherever they may be located.

The RDE provides technologies or services that fully support geographically distributed concurrent design, development and the electronic exchange of product information. The RDE commercially available high-speed communication services allow for linking to geographically diverse sites. Since team members represent different companies, organizations and product development disciplines, the RDE must be able to support whatever tools are used in a heterogeneous computing environment.

The ENTIRE concept is defined as RDE software which is comprised of a RASSP-supplied domain-independent set of infrastructure services coupled with a user supplied set of domain dependent design automation tools. Installations of the RDE will be tailored to include integrated tools and libraries which will exchange data via standard/common formats or through translators.

Another significant aspect of the RDE is a distributed database containing all of the product life-cycle data (product and management data) for both current and previous RASSP designs, as well as modular building blocks for design reuse. Each project has a different set of requirements for product development and therefore a unique set of CAD tools are needed to support the project. Recently initiated projects may utilize existing tool sets, current versions of tools, or the best possible tool solutions from multiple CAD vendors. A major advantage of this approach is that the integrated database produces, in one location, all of the relevant program documentation. This has a major positive impact in reducing the life cycle costs of a program. It also saves both the contractor and the government money in the development cycle. In order to adapt to different or changing tool sets the RDE must have the flexibility to be tailorable. The RDE will be delivered with a set of core RDE utilities plus an integrated set of CAD tools. Each site configuration of an RDE will probably contain a different set of CAD tools which have been determined by product development requirements. The CAD tools in the RDE will be integrated together along with component and model libraries. The information that is exchanged between tools will be in a manner such that tools can be substituted with different tools of at least the same functionality and still allow the data to automatically flow from one process step to another.

This ENTIRE tailoring will contribute to 4X improvement by allowing new and improved design automation tools to be utilized in conjunction with the core utilities. All of the components in the RASSP Engineering Database (REDB) are designed to allow remote team members to work together on the same project as if they were in the same room. The utilities and tools will be integrated in a manner which allows information to be exchanged securely between team members that work in remote locations (illustrated in Figure 4). One of the RDE features that supports this is the database communications. **Each RDE connects to a RASSP Engineering Database server.** The server can be hosted either on the local computer, or a computer in a remote network. Each RDE server has the ability to communicate with other servers. If data is requested from a database, if the data resides in a remote database, the data will be requested from the remote server. Secure data sharing and conferencing will be achieved with commercial encryption and decryption products. Additional security is being addressed with firewall boxes. The combination will provide authentication (no IP spoofing) and privacy (encryption).

2.4 RASSP Demonstrations and Benchmarks

The Demonstration is a key part of developing the RASSP process and tools. **By developing actual hardware and software systems,** the demonstration assesses the usefulness and performance of the RASSP design environment and its associated tools. It provides a measure for design complexity and process maturity and allows us to quantify progress toward the four-fold improvements which are the primary goals of RASSP. The demonstration, then, allows us to test the methodology and the RDE, and it lets us demonstrate the Model Year Upgrade concept.

The development of a real-time infrared search and track **(IRST)** processing system serves as a real-world application of the on-going development of methodologies and processes intended to achieve a four time improvement of cycle time at the end of the four year program. This development is an integral part of RASSP program. Real world applications are used to validate and provide metrics on the effectivity of the methodology and processes.

The specific demonstration in the Sanders RASSP program involved upgrading the Infrared Search and Track signal processor on the ARPA AIRMS aircraft for Model Year 0. This Model Year demonstration has been successfully completed. Model Year 1 upgrades the IRST processor on the F-14D aircraft. Much of the work from Model Year 0 is being reused. In the demonstration Model Years we build Virtual Prototypes using multi-leaf VHDL models and software whose source code is written in Ada. This prototype is used to support three Model Year Upgrades (0, 1, and 2). Model Year 2, featuring full 3-D capability IRST inside the original 2-D physical, weight, power, and connection constraints will fly on an F-14 in 1997. Model Year 0 is flying in 1995 on an ARPA special mission testbed aircraft. Model Year 1 (F-14 flight hardware) will be operational in early 1996.

"Our team has demonstrated that we are more than halfway to our goal by demonstrating a speedup and cost reduction of 2.2 X - 2.7 X on a real DoD weapon system upgrade."

2.5 RASSP Proliferation Approach

The primary purpose of the Proliferation function in the Sanders program is to successfully transition our RASSP process to other organizations. This team also interfaces with the Educator/Facilitator and with standards organizations such as CFI, SAE, and IEEE. They also help coordinate the activities of the University and Industrial BAA winners with the Sanders RASSP team. An additional important task of the Proliferation activity is to establish and support the beta sites as our RASSP process moves to new users. This encourages the “new users” to be part of our team and helps assure that the commercial software is useful -- there is a lot of power inherent in a large group of product buyers.

3. Accomplishments

3.1 The Road to 4 X -- Over Half Way There

The RASSP Program has ambitious goals: **4X decrease in product development cycle-time, 4X decrease in life-cycle costs and 4X increase in product quality.** We assess in this section the progress of the Sanders RASSP towards achieving 4X. We also show that the results of a balanced set of activities lead to our improved ability to perform on real problems with immediate benefit to our military readiness. This concrete, demonstrated performance on real problems provides convincing evidence of our progress. **Our team has demonstrated that we are more than halfway to our goal by demonstrating a speedup and cost reduction of 2.2 X - 2.7 X on a real DoD weapon system upgrade.**

The Sanders, Motorola, Hughes and ISX RASSP Team is developing a balanced set of approaches to address each of the factors that contribute to improvement and the barriers that impede improvement. This balanced approach involves developing new engineering and business processes and new technology as well as improving access to resources and information.

3.2 Prototypes and Corporations -- The Two “Virtual Virtues”

3.2.1 Virtual Prototypes

Virtual prototyping in Sanders’ RASSP Process **uses VHDL and Ada** and allows each contemplated design alternative to be captured in executable requirements and an executable specification; this automatically provides concurrent constructs of design documentation: provides executable design-to-baseline for the next level of design; facilitates rapid impact assessments and design verification; and provides the potential for automatic verification of design properties, alternatives, and synthesis.

The RASSP Process, being developed by the Sanders Team, applies rapid and virtual prototyping using state-of-the-art design, development and simulation tools, and DOD standards, including **VHDL and Ada, to derive early insight into product performance and risk**, and to project affordability, manufacturability and sustainability. Early application of subsets of this process, such as in Benchmark efforts, have demonstrated the possibility of applying VHDL and Ada for Virtual Prototyping of design in order to identify and rectify problems before processing the actual hardware. This approach also enhances risk analysis and mitigation by providing for generation and analysis of alternate design solutions in the virtual prototype phase rather than after “metal is bent and circuits fabricated.”

Virtual prototyping in our process will eventually allow each contemplated design alternative to be captured in executable requirements and in an executable specification; this will also provide a concurrent construct of design documentation, along with executable design-to-baseline for the next level of design.

3.2.2 Virtual Corporations

The Sanders RASSP team uses a **Virtual Corporation** concept for developing VHDL descriptions of weapon system signal processors (Figure 3). This work spans four companies and several universities and small suppliers across the country and involves all aspects of a project, i.e. design, analysis, fabrication, and test. This section describes the communications and coordination methods developed for use across the Virtual Corporation, including a brief look at the processes needed to support a distributed design database, source code coherence across multiple networks, and secure communications. The result of these efforts is a team able to complete all its modeling and designs without a single co-located designer or design review and with successful completion of all the hardware development and savings in travel costs.

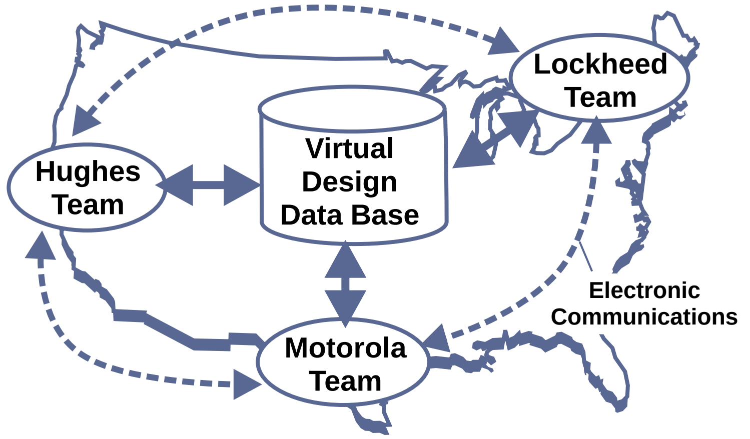


FIGURE 3. Development of the RASSP Demonstration Virtual Prototype took place in a distributed environment using cross-country electronic communications and a shared design database.

Early in the Sanders proposal effort, the goal of establishing and **operating as an integrated Virtual Corporation** was set. Since that time, the team has worked continuously to identify and refine the infrastructure necessary to reach this end. The Virtual Corporation infrastructure borrows heavily from the successful experiences, methodologies, and tools used and developed by each team member over many projects and many years. These components have been melded together with additional services and utilities identified by the RASSP team to provide the support necessary for managing, running, and succeeding within a Virtual Corporation. This effort has been extremely successful to date. The methodologies, tools, and standards that enable a Virtual Corporation span the areas of Program Organization and Management, Advanced Concept Engineering, and Data and Information Management. The Virtual Corporation infrastructure is being captured and refined within the RDE and Process Development efforts themselves so that, ultimately, the RDE may be used by the team to manage and support its continuing RDE development.

The notion of a virtual company is a fairly recent development that appeared with the advent of global networks such as the Internet. The RASSP program uses this concept to conduct business that requires a design team for the development of a complex digital processor. The technical domain is challenging because of the amount of detailed information exchange required by the design team. **The proper level and degree of information exchange is the design aspect that poses the biggest problem for a virtual company** because of the intangibles involved in creative tasks. The technical domain also has demanding resource requirements for creating and exchanging associated data in terms of workstation and network capacity. The RASSP Program challenges technology limits and explores the development of complex multi-processor digital systems with a **distributed design team.**

Integrated Product Development teams **have all of the disciplines needed to accomplish product development from concept to field support** working as a single integrated team to efficiently and concurrently create new innovative products. The team approach enables tight linkages between hardware, software, product design, manufacturing, procurement, reliability, maintainability and supportability to be established and maintained. IPD can be made significantly more powerful with the addition of tools and processes to enhance situational awareness.

Virtual corporation technology extends the concept of IPD to encompass multiple companies, geographically separated to perform as if they were a single company located in a single location. Virtual corporation technology allows the flexible creation of teams comprised of electronically co-located workers and addresses both engineering and management issues.

The Sanders Team employs a variety of techniques and tools to enable an integrated program management approach. None of these is more fundamental to our virtual enterprise than that of the Integrated Product and Process Development Team. This methodology focuses on the use of interdisciplinary teams throughout the process and product development cycle. This philosophy provides for a broad range of expertise from a range of disciplines when conducting each step in the development process. In the Sanders virtual enterprise, these teams are further diversified by ensuring that each IPPDT has membership from each corporate enterprise member.

Visionary Demonstrations:

One of the primary difficulties involved in working in a virtual enterprise is the coordination of technical activities. The identification and proliferation of a system concept within a geographically distributed team is key.

The RASSP team has employed the concept of a **"Visionary Demonstration" (Visdem)** to help satisfy these needs. A visdem is a multimedia application developed and refined in a rapid prototyped manner. This visdem typically captures program organization, technical approach, and most importantly, the operational concept. This operational concept constitutes a visual contract between development team members and the target user community, showing how the system under development is to look, feel, function, and be utilized.

Conceptual prototypes:

The next step in the validation and development cycle of the RASSP system is to take the most promising concepts identified in the visdem and build them in a much more realistic environment. This step is the development of a conceptual prototype.

The conceptual prototype varies significantly from the visdem. First, the **conceptual prototype is built on the development/delivery platform.** It is developed using the language and standards identified by the core development team. It uses and provides reusable code synergistically with the core development team. It provides some very limited functionality.

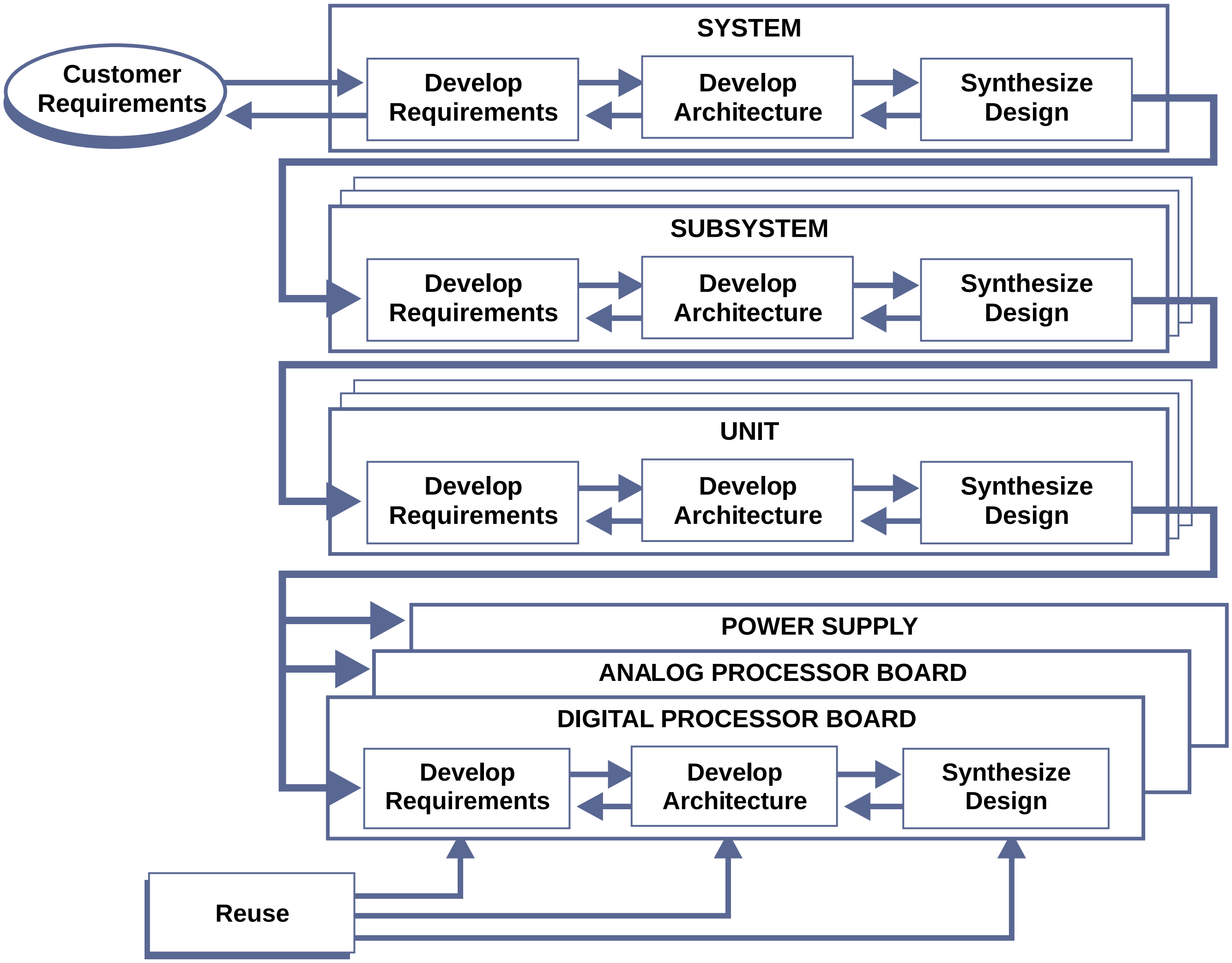


FIGURE 4. The RASSP Design Process allows the top-down evolution of system requirements through an iterative decomposition of requirements into smaller units.

Its major focus again is the illustration of functionality rather than the robust provision of that functionality.

RASSP inter-company network:

The primary purpose of the RASSP inter-company network is to support the work on the RASSP program by providing electronic communication between team members. A variety of computer platforms and operating systems are employed on the network. The RASSP inter-company network allows for information transfer among these heterogeneous systems in the form of Email, application data files, Video Teleconferencing (VTC), and shared windows.

3.3 Top Down Design and VHDL -- The RASSP Process Twins

3.3.1 Top Down Design and VHDL

Top-down VHDL combined with structured software development enable modular product design. Top Down Design begins with development of VHDL models for entire systems and continues on to hardware/software partitioning and through detailed hardware and software design and development (Figure 4). These models comprise the Virtual Prototype(s) of the system and system elements.

The implementation of the Virtual Corporation allows designers instantaneous access to data and highly interactive and dynamic electronic communication with all design team members. While the technologies to support such goals are still maturing, the RASSP team took a realizable approach by implementing design data bases and technical communications with state-of-the-shelf hardware and software. **The cornerstone of the hardware approach is IEEE standard VHDL.** This choice is based on the interoperability and design documentation attributes of the language. These features allow designers to share design objects and convey a working understanding of design behavior, enabling the creation of a virtual company.

RASSP uses the characteristics of HDL languages and Virtual Prototyping as the basis for a methodology that produces a top down design approach that includes a totally portable

VHDL description. This also includes modeling methods for discrete levels of abstraction that facilitate interoperability. These features are ideal for a multi-company design project that must accomplish virtual system integration and maintain a design data base.The methodology requires a system of procedures for data management across the virtual company. A data promotion scheme was developed that allowed incremental levels of dependency and maturity for source code.

3.3.2 Reuse

Our RASSP Team recognizes the importance of reuse and reuse libraries. We are capturing the VHDL elements from the IRST Demonstration Model Year 0 and from Benchmarks 1 and 2 in a database and will be demonstrating their reuse in Model Year 1 and Benchmarks 3 and 4. Our experiences will be valuable in assessing the additional requirements for easy reuse. We are also demonstrating how processes can be reused by applying parts of the Model Year 0 process to Model Year 1 and to Benchmarks 3 and 4. Because we learned a great deal during Model Year 0 and because Model Year 1 has added complexities such as legacy system considerations, the process has been refined and tailored to meet the needs of the next cycles of designs.

Application-specific systems benefit from reuse of design information and functional block libraries from past designs. Algorithms can be rapidly designed using reuse libraries of commonly used functional blocks. Architectures can be quickly synthesized from reuse components of past designs. Use/reuse of the library of primitives allows the engineer to rapidly and confidently capture system functionality in terms of behavior, variables and communication channels for data/information flow among system elements. Critical paths for control and data flows can be identified, captured and analyzed, and requirements can be associated to, and linked with, functions. And the biggest payoff is that there are no irrevocable decisions, so it is not necessary to get it “perfect” the first time -- one just has to improve it each time through. Thus the inevitable “misstated requirement” or misunderstood operational environment is no longer the largest cost driver in the design and fielding process.

3.4 Progress Through Process and Method

The Sanders RASSP Team’s evolution of the RASSP Process began at program initiation. Since then, Integrated Process and Product Development Teams (IPPDTs) for Demonstration and Benchmark performed design of systems using such of those process attributes as were applicable. For example, the Demonstration and Benchmark tests are, in and of themselves, applications of the Model Year design concept; VHDL and Ada were used to develop and evaluate virtual prototypes before construction of actual hardware. After these efforts had been underway, representatives of each team were formed into an ad hoc Process Focus Team that captured what had been done, how it was done, identified strengths and weaknesses, and began to document and refine the process. This documentation and refinement is currently well underway, under the Systems Engineering IPPDT’s Process Group and will lead to use of the defined process in the upcoming Demonstration and Benchmark applications, as verification that the process has been adequately captured and as validation of process effectiveness. Throughout this effort, the RASSP Design Environment (RDE) IPPDT has been tracking the progress and looking at how to apply automation to the process environment by incorporating tools, utilities, databases and communications so as to enhance performance of the design effort.

At the start of 1995, we presented a model for reaching 4X. This model is based on the results of a task analysis of the design process from very early system concept development and feasibility through development, test, production, and field support. Approximately 70 specific tasks were identified, and the associated duration, based on current practice, for each task was determined. The baseline development timeline resulting from the assignment of duration to tasks was compared with the current practice model proposed by Vijay Madisetti and Jack Corley of the RASSP Education and Facilitation team and found to fit easily within their minimum and maximum timelines. **This favorable comparison helped increase our confidence that we had accurately captured the basic product development process.**

Key achievements in Sanders’ RASSP Process.

- The process is captured in electronic form in our RDE at differing levels of abstraction, wherein the top level is a tool-independent graphical view, and lower levels provide discrete tool-dependent workflow fragments.
- The on-line process description includes process step descriptions, entry/exit criteria, tools, metrics, performance estimators (“thermometers”), guides and aids.
- The top-level process application is tool- and workflow-manager software-independent.
- Tool integration is achieved by the use of the RDE’s “ENTIRE” application. The process structure is tailorable to the specific needs of a project or customer, and phases and levels can be added -- as needed -- and integrated into the overall process.
- The process accommodates the spiral engineering model by supporting rapid iterations through requirements, architecture and synthesis and incrementally developing the product with rapid top-down iterations and bottom-up feedback.
- The use of technology-independent functional models for the virtual prototype enhances reuse of functional primitives, and allows architectural trades to be performed more rapidly.
- The virtual prototype can be tightly linked to the synthesized design through functional-to-structural interface models, thereby tightly coupling all levels of the physical design to the functional behavior.
- The process builds the product through a common functional description for hardware and software.
- This process supports the concept of “model year upgrade” because it provides the use of previous model year models as a baseline for further developments, allows

for modification of the functional models in the virtual prototyping stages, and allows for partitioning and re-targeting during the synthesis activities.

The RASSP Process Now:

To present the RASSP Process in a "recognized engineering format," the Sanders Team chose to adapt the preliminary version of the **IEEE System Engineering Process, P-1220 ("Standard for Application and Management of the Systems Engineering Process")** for tailoring into the RASSP Process Description and to apply the **IDEF 0** Format for the design construct representation.

The Sanders Team has been defining and documenting its common description of the "RASSP Process" for the purpose of achieving both commonality of understanding and "process configuration management." Further, they recognize that at contract end the legacy process definition must be sufficiently robust to provide for its continued successful use by industry and government engineering design teams.

The Sanders RASSP Process is intended as a combination of "top-down" hierarchical and "spiral development" engineering sequences, applied in Model Year progressions. It has been developed using the preliminary version of the new IEEE 1220 standard for systems engineering to assist in definition and description. Our overall process description extends through the six phases of the system life-cycle of engineering effort related to product development, down six levels of product decomposition, and includes three recursive steps within our "Process Engine" for analysis: "develop and validate requirements baseline," "develop and verify functional architecture," and "synthesize and verify hardware, software, and physical design."

Sanders' process uses a "finish-to-finish" engine rather than a "finish-to-start" sequencing. **Rather than require each activity to finish before the succeeding one starts, they only require that all previous activities be complete before succeeding activities can terminate.** The result is encouragement to evaluate details and mitigate risks early in the design. The Sanders RASSP Process also encourages rapid prototyping activities, including early prototypes of user and other interfaces and of partial and end-to-end threads through the design to permit independent evaluation, optimization and validation. The Process structure is tailorable to the specific needs of customers and projects, so the specific steps within any particular iteration of its application can be different from all others. However, the overall methodological approach should be consistent.

How well is all this working? **First Criterion: Has the Sanders RASSP Process been used? Answer: Yes,** multiple new weapon system upgrades have been and are being started. **Second Criterion: Are the team's parent companies incorporating the RASSP Process? Answer: Yes,** each of the three major industrial members are internally instantiating the RASSP Process.

3.5 ENTIRE

Data Base Access and Data Control – The Key to Reuse, the Focus of Automation

The RASSP Design Environment (RDE) is central to our data access, data integration, and automation efforts. **The current version of the RDE is the fourth build of the RDE out of a planned total of ten, and represents a total of 300,000 lines of code developed for this program.** The RDE development effort uses previously developed RDE utilities in conjunction with the RASSP process of software development which is rated at SEI Level 3 (Repeatable and Transferable). This release of the RDE contains a common set of infrastructure services for use in a wide range of applications: **common desktop, automatic metrics collection, metrics analysis, reuse utility, technical review utility, problem reporting utility, log utility, and remote data access.** The RDE is being tailored for use on our Model Year 1 demonstration to include domain specific tools, translators, libraries, and process support. A preliminary list of applications for the Demonstration are given in the paper. The fully populated and tailored design environment is described as the ENTIRE concept (ENvironment and Tools for an Integrated RDE).

The RASSP Design Environment (RDE) facilitates Integrated Product Development (IPD) by providing a collaborative work environment. The RDE provides support for automating the product development process. The RDE enables the IPD philosophy with its support of **rapid iterations, incremental promotion, and scalable configuration management controls.** A fundamental thread in supporting IPD is communication between individuals within and across teams of people, especially when the teams are not co-located. The RDE provides technologies or services that fully support geographically distributed concurrent design, development and the electronic exchange of product information. The RDE supports whatever tools are needed in heterogeneous computing environments. The ENTIRE concept is defined as RDE software which is comprised of a RASSP-supplied domain independent set of infrastructure services coupled with a user-supplied set of domain dependent design automation tools. Installations of the RDE will be tailored to include integrated tools and libraries which will exchange data via standard/common formats or through translators. Another significant aspect of the RDE is a distributed database containing all of the product life-cycle data (product and management data) for both current and previous RASSP designs, as well as modular building blocks for design reuse.

ENTIRE Can Be Tailored

Each project has a different set of requirements for product development and a unique set of CAD tools needed for support. In order to adapt to different or changing tool sets the RDE must have the flexibility to be tailorable. The RDE will be delivered with a set of core RDE utilities plus an integrated set of CAD tools. The CAD tools in the RDE will be integrated together along with component and model libraries. This ENTIRE tailoring will contribute to 4X improvement by allowing new and improved design automation tools to be utilized in conjunction with the core utilities.

ENTIRE Uses SHORE/EXODUS (A University Developed and ARPA Contracted Database).

The RDE stores and retrieves project and product information from the RASSP Engineering Database (REDB). This heterogeneous database contains all the information required by the project, from lists of users to design schematics. The REDB is distributed for team members to be able to securely access project and product data from remote locations. Information is placed into the REDB through a programming interface. This interface allows utilities or tools to access the repository independent of the underlying database. There may be only one, or multiple databases beneath the interface. Each database must implement the functions in the interface to work with the RDE. The programming interface allows access to object-oriented databases, relational databases, or CAD framework databases. The REDB is the repository for the design information that is commonly shared between the tools. The information is stored in a common format for each type of design data. Any tool that requires the data as input can then acquire and use the data.

Tool Encapsulation & Integration Not Needed

This scheme does not use tool-to-tool translators, but format-to-tool translators. The advantage of this technique is that if a translator is required, it only needs to read and write the data from a tool into a common format, independent of other tools with which it interfaces. This technique, based on CFI's concept of Design Representation (DR) solves the tool-to-tool interoperability problems. The formats are then readily captured by databases.

ENTIRE Allows Distributed Development

All of the components in the RDE are designed to allow remote team members to work together on the same project as if they were in the same room. The utilities and tools will be integrated in a manner which allows information to be exchanged securely between team members that work in remote locations.

What's In ENTIRE?

Desktop

The RDE Desktop is an environment shell in which working conditions can be customized and tools can be encapsulated, accessed and launched.

Remote Data Access Utility

The RDE Remote Data Access Utility (RDA) is comprised of client/server software for accessing the RASSP database. This utility consists of a Service, a Server Broker, and a Client. These database operations are implemented using the Team Design Manager (TDM) tool by Cadence.

Log Utility

The “Log Utility” provides a mechanism for people to enter miscellaneous text information about things they are doing on a day-to-day basis.

Metrics

The RDE automates data collection and metric generation as much as possible. Towards this end, the RDE metric tools query the various databases encapsulated within the RASSP Design Environment and extract necessary data. Then, the metric tools collate the data and create various metric reports.

Problem Reports

The RDE provides a problem-reporting mechanism for design teams to effectively capture “Problem Reports” and to distribute those reports to the responsible individuals. It has a Motif-based user interface for the GNU problem report tool.

Reuse Utility

The Reuse Utility encourages the reuse of software modules and structures by providing a method of storage and retrieval of information on available reusable units. A series of friendly, organized GUIs assist the user in adding, updating, and querying the database.

Technical Review Utility

The Technical Review Utility facilitates Peer Reviews at each stage of a development process to help ensure a quality design. The RDE Technical Review Utility allows the identified reviewers to review the design information when they have the time to do so. The Review Utility removes geographic and temporal constraints from the review process.

ENTIRE Supports Multiple Tools

The RDE can be used throughout all phases of a product's life cycle from conceptual and detailed design through production to field support. A wide variety of disciplines will be utilized throughout the product development process which requires the use of many classes of tools including tools for program and project management, requirements capture and analysis, algorithm development, software engineering, and electrical and mechanical hardware design, modeling, and simulation. The RDE software currently runs on Sun SPARC platforms using SunOS version 4.1.3. TDM (Team Design Manager) from Cadence is required for source configuration control of design data.

Validation of ENTIRE

Validation of the RDE will be done with the Demo team's Model Year 1 design (Figure 5). The validation will include test of the flows of information between tools, making sure the libraries work properly with the versions of the tools and ensuring that the tools are properly installed in the desktop, and that the necessary tool-specific TDM policies are written and work. Validation of the RDE also involves regression tests to ensure the software conforms to the requirements.

Summary

The ENTIRE Concept supports a heterogeneous computing environment, links between geographically diverse locations, tailorable configuration management, and exchange of product information between the many varied disciplines. ENTIRE supports an improved product development process allowing for rapid iterations, incremental promotion, and scalable configuration management controls. In this role ENTIRE is an enabler to help achieve the RASSP goals of 4X improvement in product development time, cost, and quality.

3.6. RASSP Takes Flight

3.6.1. AIRMS and F-14 IRST

The development of a real-time infrared search and track **(IRST) processing system now flying on the ARPA Advanced Infrared Measurement System (AIRMS)** program served as a real-world application of the on-going development of methodologies and processes intended to achieve a four-time improvement of cycle time at the end of the four-year program.

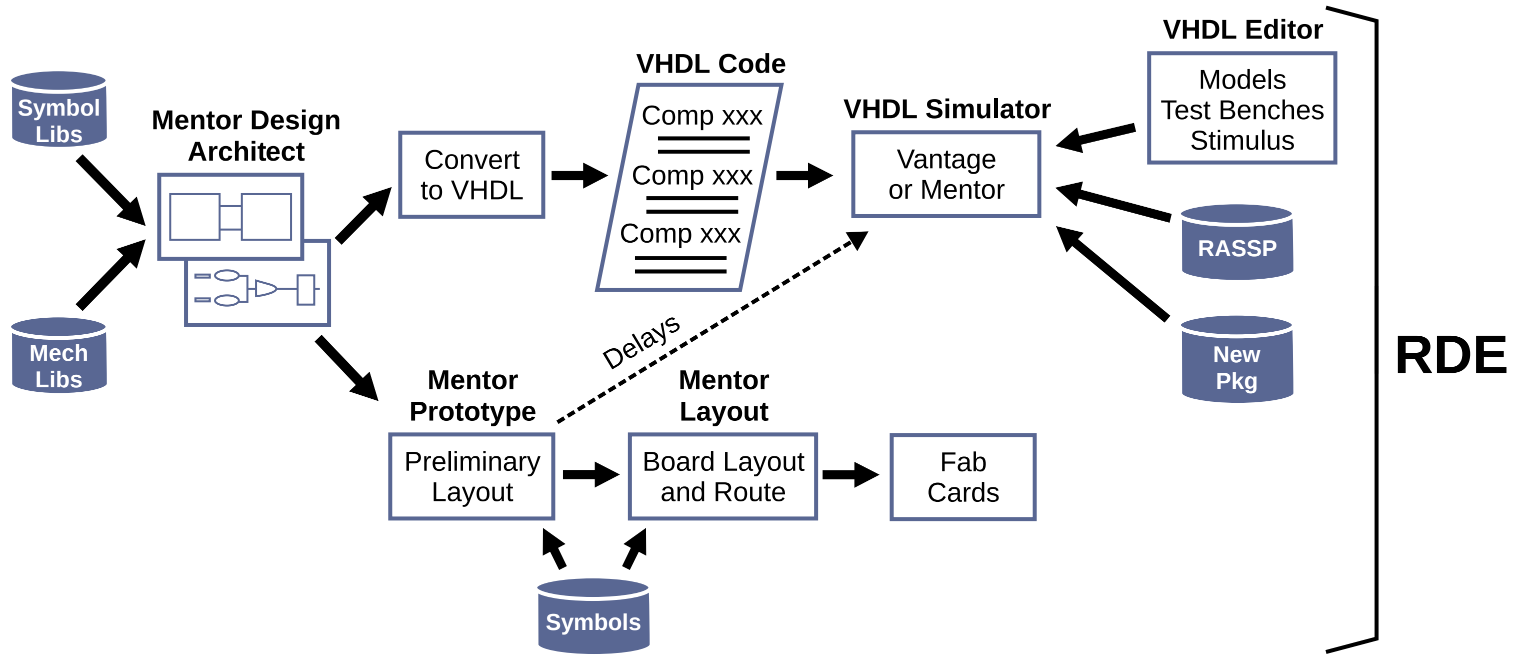


FIGURE 5. The RASSP Demonstration Model Year 1 detailed design process will be supported within the RDE using COTS tools.

We created a complete VHDL virtual prototype of processing hardware and Ada software before the design was fabricated. The rapid laboratory checkout that ensued is an indicator that virtual prototyping has great value. The virtual prototype was used to simulate processing modules, custom interface modules and the Ada software. The RASSP IRST project was performed by a virtually co-located team with Team members from Hughes in El Segundo, California; Motorola in Scottsdale, Arizona; and Sanders in Nashua, New Hampshire. Using the Internet for file sharing, e-mail, and video teleconferencing, the entire hardware and software design was performed without requiring travel for design reviews or coordination. VHDL descriptions done at each location were integrated to form the virtual prototype. The virtual prototype facilitated distributed design checkout since the designer and reviewer could check their own portion of the design from their own office.

IRST Processing System

The virtual prototype developed for RASSP models a complex multiprocessor system composed of commercial off-the-shelf (COTS) processing modules, custom interface modules, and Ada code. This section describes the IRST processing system elements in order to aid in understanding the scope of the virtual prototype.

Infrared Search and Track processing detects unresolved (sub-pixel) moving objects in an infrared image. Performance is limited by scene clutter (clouds and terrestrial background). Algorithms are applied to register multiple frames of data, filter out clutter, boost target signatures, and thereby detect and track targets. The detected targets are displayed with graphic symbology overlays on top of the original scene data.

The design can handle either a standard video input stream or a custom 135 Mbyte/second digital data sensor stream. The video output displays sensor imagery with graphic symbology overlays. Four custom modules were designed for interfaces. COTS modules were used for the signal processing and host controller. **The IRST processing system virtual prototype totals over 39,000 lines of VHDL. An additional 18,000 lines of software implement the algorithms and control software.**

The Data Input/Distribution module is a card with 35 ICs, including 4 FPGAs. We used two daughter cards to adapt to sensor-specific electrical and timing interfaces. A custom interface is used for control and status. A Mercury RACEWAY interface permits high-speed video transfer to the processor modules. A video output displays one of the two sensor inputs with symbology. A video crossbar connects the two daughter cards, the video output, and the RACEWAY interfaces. Routing logic under software control passes the image to selected processing elements in the multiprocessor system. The IRST processing system soft ware contains 18,000 lines of code.

Virtual Prototyping — The Error Sieve

The RASSP virtual prototype was developed using a top-down VHDL design methodology with progressive addition of more hardware details. The process supports hardware/software co-design, with the initial phase of the virtual prototype serving merely as a performance model of the end system that shows busses, major computing elements, and I/O. We modeled software and sensor workload as tokens to evaluate processing element and bus loading. In subsequent design refinement, we developed behavioral models of processing elements, interface circuits, and buses. These, in turn, were refined to register transfer level descriptions that supported design synthesis of programmable logic. Instruction set level modeling of the processing elements allowed execution of control and built-in test Ada software within the VHDL model. By completing each level before beginning the next lower level of detail, we caught design errors early in the design cycle.

Instruction Set Architecture Model

A VHDL-Based Instruction Set Architecture (ISA) model of the Intel i860 microprocessor was developed by the Georgia Institute of Technology. The ISA model implements all registers, instructions, and status logic visible to a programmer. It allows i860 object code to run as it would on real hardware.

System Level Virtual Prototype

The system level virtual prototype combines all system hardware and software elements. The simulation checks for consistency of protocols across interfaces: board-to- board and hardware-to-software. The simulator also aids development of low-level diagnostic tests debugged on the virtual prototype and then executed on the real hardware. All tests were written in Ada.

Results

The simulation time for a large complex processing system such as the IRST system can be very long. During November, we initiated 169 simulations, of which 135 were completed. They simulated 897 milliseconds and used 402 hours of wall clock time running on a Sparc 10. By running Ada code on the virtual prototype, we discovered three hardware errors and eight software errors. Checking out the design on the virtual prototype helped us discover numerous design errors; however, the rate of discovering and fixing errors was slow due to the current long nature of simulation run times.

All hardware designs checked in the virtual prototype worked in the laboratory the first time (Figure 6). Some analog portions could not be checked, and these required minor tuning. The virtual prototype provided a forum for hardware and software engineers to discuss details sooner and change early system concepts when performance simulation or early VHDL modeling showed timeline problems. Many software errors were fixed before the laboratory checkout. Running the actual software on the ISA model identified hardware design problems not discovered in standard VHDL simulation. Simulation run times were so slow that we could explore only those activities near the beginning of the hardware initialization cycle (the first 150 milliseconds). We could test only a limited portion of the software because of the slow simulation times and because the operating system could not be run on the virtual prototype; only software that did not use operating system calls or the run time system could be executed. There is a clear need for better simulation run times. **Executing the operating system and run time software is essential to fully check out the software and refine the hardware/software design.**

The virtual prototype changed the development schedule such that although the **design stage lasted longer than in conventional development, the hardware checkout and system integration went much faster.**

3.6.2 SAR Image Processor for UAV

A virtual prototype of a synthetic aperture radar processor is being created by Sanders as part of Benchmarks 1 and 2. It includes VHDL models and Ada application code. The current status is that the virtual prototype has been completed, and we are developing a hardware prototype to validate the virtual prototype.

Our RASSP Benchmark 1 is phase one of a two-phased development of a synthetic aperture radar (SAR) processor. The Benchmarks are another set of means by which the government assesses the progress the contractor is making toward program goals. RASSP chose the SAR processor as the initial benchmark because it is a computationally challenging system that is a good test vehicle for architectural modeling and virtual prototyping. The intent of RASSP Benchmark 1 was to obtain metrics on the developer's initial design process, and to start developing the metrics for comparing the usefulness of design processes and changes to those processes.

SAR Processor Requirements

The requirements for the SAR processor were specified by the government’s designated benchmarker, MIT Lincoln Laboratory. They supplied the requirements for the application utility, the signal processing algorithms, data timing and formats, and the physical requirements of the hardware. The SAR processor is a particularly computationally intensive application in that it requires many complex FFTs and vector multiplications.

Design Process

The design process was one of virtual prototyping: implementation trade-offs, followed by modeling, software development and some hardware design. VHDL was used to create an end-to-end virtual prototype model of the SAR processor. The hardware design went to the layout level to insure that the virtual prototype could indeed be realized in the allocated hardware configuration.

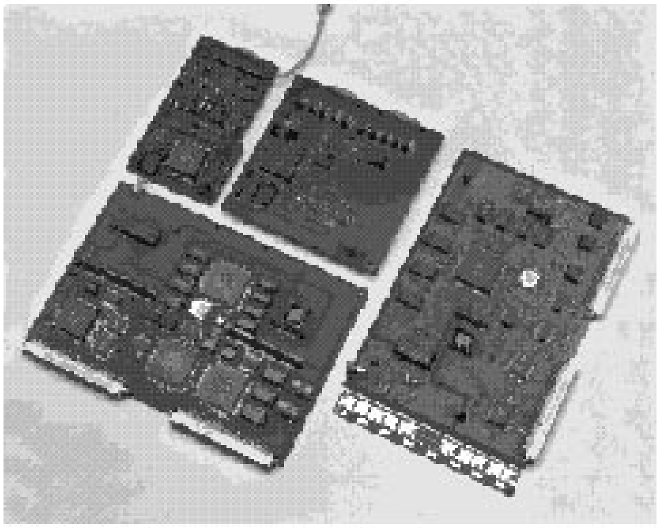


FIGURE 6. The RASSP AIRMS Demonstration custom hardware was checked out in the virtual prototype before fabrication, leading to significantly reduced integration and testing.

The first step in the virtual prototype design process was to evaluate several different designs with different hardware/software configurations. There were four options evaluated for the implementation of the SAR processor before detailed modeling began. Three estimation programs were used to evaluate the top level trade-offs. A spread sheet model was developed to quickly evaluate weight and power as board count varied. Another spreadsheet model was used to estimate the development and unit costs based on typical tasks durations, lines of code, types of boards, and kinds of enclosures. Finally, the life cycle costs (LCC), including development costs, were estimated using the much more complex PRICE-H and PRICE-HL parametric cost estimation models. This methodology is significant because it allows the designer to understand all aspects of partitioning functions into various hardware/software combinations.

Virtual prototype modeling and the VHDL testbench

The next step in the virtual prototype design process is at a more hardware oriented, less abstract level. Chief among the benefits of virtual prototyping at this level are the enhanced communications provided between customer and designer about a new design and the implications of the customer requirements on cost and performance. As component models become commercially available, the designer will be able to rapidly identify design flaws without having to actually build and debug hardware. Further, the completed virtual prototype can serve as an archive of the design for future designers to access and use for component replacement and upgrades, without requiring the physical hardware. Lastly, the virtual prototype can serve as an “Executable Specification.” This allows a customer to exactly specify, at any desired level, the requirements for what is to be built, with minimal misunderstanding or error.

The level of simulation which we wanted to achieve was that the virtual prototype should be identical, from a software perspective, to the physical prototype. We modeled the interfaces and functions of the key elements so that they could be controlled by the same application code being created for the hardware. This approach provided the co-design between software and hardware which was desired.

MIT Lincoln Lab created a VHDL Executable Requirement. This provided an input data set, a VHDL model of the input and output source and sink, and a simple VHDL behavioral model of the SAR processor. The virtual prototype developed during the benchmark was to be tested by substituting for the behavioral model of the executable specification.

VHDL Modeling of the SAR Processor

Fifteen VHDL models were developed for the SAR processor. At the top level, the executable specification test bench was used. Lower level test benches were used for different component sets. The elements chosen for modelling represent key pieces of the system that future designers will need to rapidly implement model year upgrades. Many of these models are of industry standard parts, thus increasing their reuse utility. **A total of 14,909 lines of VHDL code were written, 6,823 of which were executable. The average productivity for both models and test benches was 32 LOC/day.**

To summarize the VHDL modelling effort, models for a number of interfaces, components, and algorithms were developed. Many of these models are of industry standard components. Capturing the functionality of the various portions of the SAR processor in a virtual prototype will foster use for future designs, enabling a very quick turn around on model year upgrades. Key to our methodology was the ability to execute application software on the virtual prototype. This allowed the VHDL model and software to be integrated, tested, and corrected before hardware fabrication. As a result, physical hardware and software integration will be greatly simplified and less costly. The application software was required to be in Ada. A total of 2,301 lines of executable code were written. The average productivity was 33 LOC/day.

Benchmark Conclusions

The purpose of the SAR processor benchmark being performed was to exercise and measure a design methodology under development. The design methodology features the ability to perform trade-offs, and virtual prototype modeling features the ability to reduce development cost, add design time, and create of executable specifications for long term supportability. This process greatly reduces the post design documentation creation and validation costs of long life systems. Several key lessons were learned in the Benchmark. First, **there is tremendous value in having specifications in both text/pictorial and executable/VHDL** form. One validates and clarifies the other. The textual version provides an easy-to-understand top level description. The executable specification provides very precise algorithmic and hardware interface data. Second, **many tools and techniques with varying levels of fidelity are required to evaluate a virtual prototype.** Cost models, weight and size models, algorithmic models, and hardware models all play a role in prototyping a new system. Third, **extensive simulations are required to properly simulate software functionality.** The VHDL models for these simulations must be designed to minimize CPU time and memory usage. Last, **the VHDL system testbench must be designed to validate the system functionality** but should limit the data to be processed to a minimum. The partial image test bench provided essentially all of the validation required to minimize the risk of the design, and the full image test bench provided some additional information. The results of this benchmark will be used to fine tune the RASSP approach, providing the government and the prime contractor with valuable information about the ability of VHDL to ensure hardware and software integration and to validate system performance before hardware fabrication. In addition, **the models and the hardware design have been requested for use in several new programs, including Benchmarks 3 and 4, a ground penetrating radar, and an upgrade to a fighter radar.**

3.7. Conclusion -- Where Are We on the Road to 4 X?

Comparison of Current Practice Models with Measured Results.

Analysis of the program results in three important conclusions: **achieving 4X requires more than within-task cycle time reduction; the early phases of the product design process are shortened the least; while the later phases show the greatest**

benefit; and the data show that a three times improvement in cycle-time can be expected by applying RASSP improvements to individual tasks. Achievement of the full four times improvement requires integration of individual tasks using the RASSP Rapid and Disciplined process to achieve effective task concurrency.

Some phases of the product development process are accelerated a great deal while others remain nearly the same in duration. Preliminary Design Phase takes nearly as long with RASSP as without. This is because the use of RASSP Top Down Design methods and Virtual Prototyping demand more work prior to PDR than the traditional methodology. **Detail Design Phase is substantially reduced** because the Virtual Prototype has matured the design significantly. **The discipline and simulate-before-build philosophy of RASSP make the E&MD Phase much shorter.** This differs from a more traditional approach that allocates very large blocks of time to system integration and the correction of errors carried from the beginning of the design process. Rework time is reduced or eliminated.

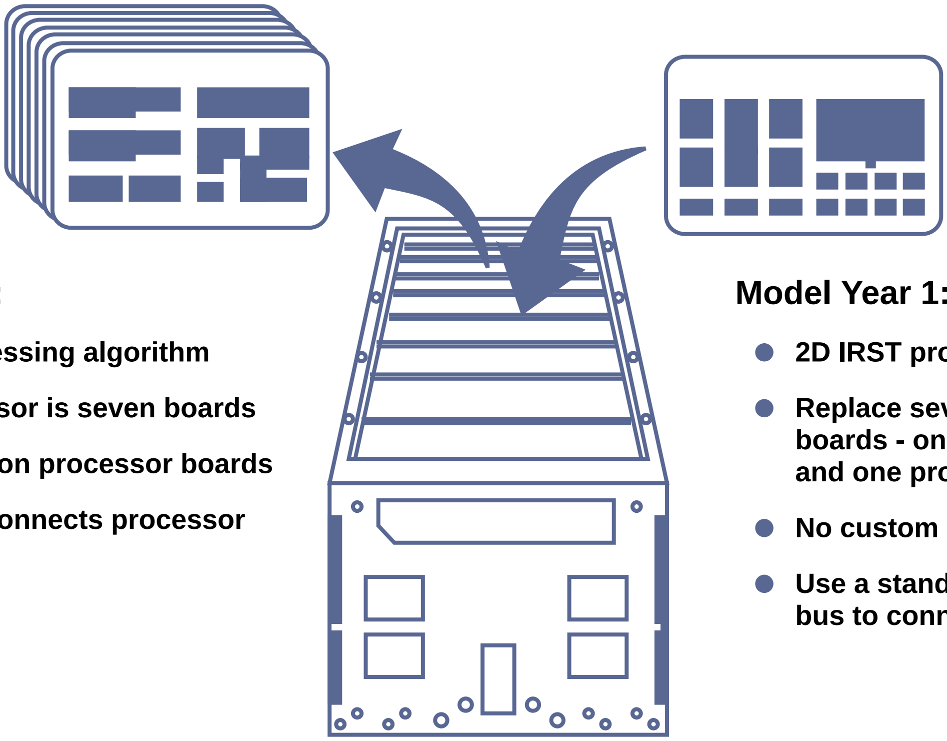
The largest contributor was Top Down Design using VHDL which also includes Structured Software Development (we use Ada) for programmable processing elements. **Reuse was an important contributor. Finally, situation awareness** was cited nearly as often as reuse and significantly more often than improved design automation tools.

The Model Year 0 IRST Image Signal Processor development undertaken as part of the Demonstration portion of the program provides a measure of how we are progressing toward the 4X goal. A comparison with the achieved **schedule and cost** of the IRST Demonstration with a similar programs and standard models reveals a range of between **2.2X and 2.7X improvement** in both. Everything that was simulated using the Virtual Prototype worked the first time. In this case rework time was eliminated. However, integration time was impacted by the need to correct errors in portions of the design that had not been simulated. Integration time was less than that typically associated with a design of this complexity. While quality, measured as fitness for use, was high when the hardware was delivered to the Aircraft, there is clearly room for the **additional improvements in quality that will lead to near zero integration time.**

4. RASSP – Future Plans

We are continually working to more completely define and optimize our development process. We have learned many detailed lessons during the process development and during our demonstration and benchmark work that has allowed us to redefine the process for future work. We will continue these efforts and will work to incorporate the experiences from our Beta Sites. In particular, we are working to capture process descriptions electronically to provide them in a database and to verify the correct connections of process inputs and outputs; we are developing process simulation capabilities; and we are working towards the incorporation of process metrics into a continuous process improvement process.

Design Reuse -- Our success in the AIRMS IRST Model Year 0 demonstration in achieving a significant speed up in the design process was driven primarily by the use of a HDL-based design



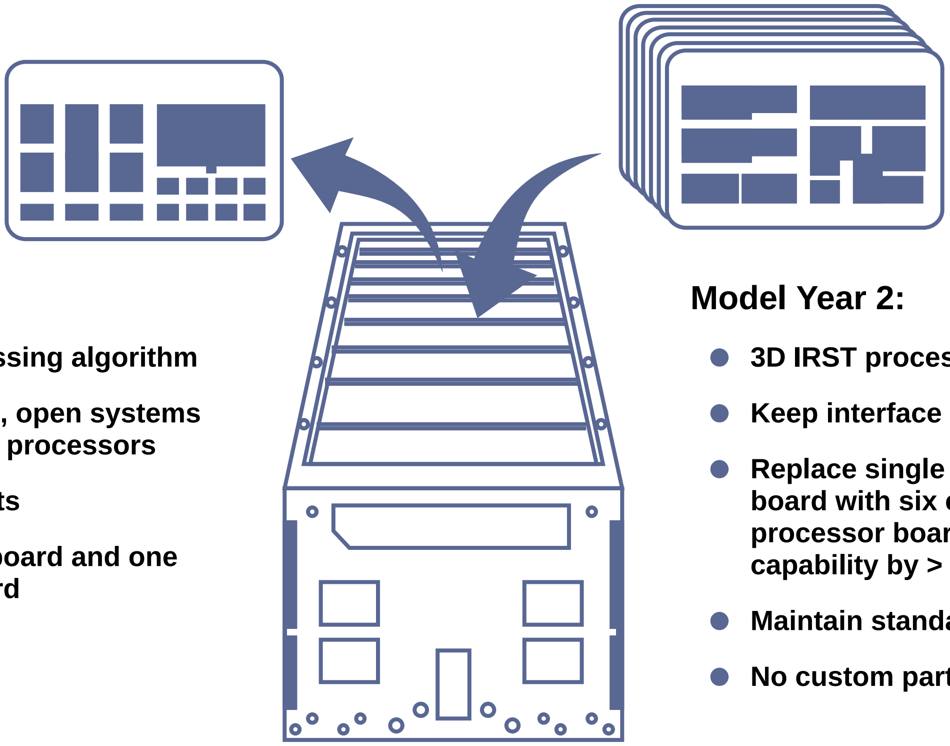
Current WRA-2:

- **2D IRST processing algorithm**
- **Signal processor is seven boards**
- **Custom logic on processor boards**
- **Custom bus connects processor boards**

Model Year 1:

- **2D IRST processing algorithm**
- **Replace seven boards with two boards - one interface board and one processor board**
- **No custom parts**
- **Use a standard, open systems bus to connect processors**

FIGURE 7. During Model Year 1 of the RASSP IRST Demonstration we will replace seven boards in the F-14 WRA-2 with two boards maintaining functionality and implementing an open systems interface that will support future upgrades.



Model Year 1:

- 2D IRST processing algorithm
- Use a standard, open systems bus to connect processors
- No custom parts
- One interface board and one processor board

Model Year 2:

- 3D IRST processing algorithm
- Keep interface board
- Replace single processor board with six copies of new processor boards to increase capability by > 10X
- Maintain standard bus
- No custom parts

FIGURE 8. During Model Year 2 of the RASSP IRST Demonstration we will use the interface built in Model Year 1 and add new processing boards with twenty times the current processing power to implement a three-dimentional processing algorithym.

process and facilitated by capabilities for remote collaboration. In determining the factors which will lead to reaching our goal of 4X the use of an HDL-based design process was key. However, an important factor in reaching the 4X goal which we have not exercised yet is design reuse. We are now entering a portion of the program in which we will be able to take advantage of design reuse and tools to support it. We are building design reuse utilities into our RASSP Design Environment and we have identified opportunities on both the upcoming demonstration and benchmark work to reuse past designs.

RASSP Design Environment -- The RASSP Design Environment has completed four builds and our first external release. During the next year we will be performing three more builds of the RDE, culminating in our second external release. Key to our efforts in this area in the next year will be completion of a vendor-independent database interface for the RDE. This will allow tools to store data in their own natural data format and for data translators to be invoked automatically if a different data format is required. This approach will speed the integration of tools into the environment and also will allow some access to tool-dependent libraries even without invocation of the tool.

F-14 IRST Demonstration Model Year 1 -- Our objective is to complete Model Year 1 of the F-14 IRST Demonstration in nine months (Figure 7). During this work we will replace seven boards in the current F-14 WRA-2 box which implement two dimensional IRST processing. We will replace these eight boards with two boards to implement the same functionality. One board will be an interface board and the second will implement the current two dimensional processing algorithm. In the process we will capture a complete system interface description and new hardware description in VHDL. We will also define a new system bus based on a standard open systems bus for use in Model Year 2.

F-14 IRST Demonstration Model Year 2 -- During the next year we will begin work on Model Year 2 of the F-14 IRST demonstration (Figure 8). Model Year 2 will implement three dimensional IRST processing in the WRA-2. We will use the interface board and the standard system bus from Model Year 1 and will implement the three dimensional IRST processing in general-purpose processor boards which will fit in the seven remaining slots. The end result of Model Year 2 will be a processor with a peak processing capability at least twenty times higher than the current IRST processor. The Model Year 2 IRST demonstration hardware will be made available to the Navy as a potential upgrade the existing F-14 capability.

SAR Image Processor Benchmark -- Within the next two months we will complete the SAR processor hardware and will have a complete system that processes the full data stream using only four slots of a VME chassis. This system will be compared to the virtual prototype which we have completed to verify the design process and the hardware.

F-15 Radar Benchmark -- During the next year we will execute our next benchmark, a radar processor upgrade for the F-15. This benchmark will replace the current front end portion of the radar with an improved system which both enhances performance and reduces cost. If successful, this hardware will have the potential to be used in over a thousand planned radar systems with potential cost savings to the government of over one hundred million dollars.

References

The following are references for documents relating to both the RASSP program in general and to Sanders RASSP Program specifically. Many of these references are available for electronic distribution:

[1] Mark Richards, “The Rapid Prototyping of Application Specific Signal Processors (RASSP) Program: Overview and Accomplishments,” Proceedings of the First Annual RASSP Conference, August 1994.

[2] Jim Summers, “Rapid Prototyping and the RASSP Design Environment,” Proceedings of the First Annual RASSP Conference, August 1994.

[3] Mike Vahey, “Lockheed's Image Signal Processor RASSP Demonstration,” Proceedings of the First Annual RASSP Conference, August 1994.

[4] Ray Dreiling, “Processes and Experiences in VHDL Top Down Design,” Proceedings of the First Annual RASSP Conference, August 1994.

[5] Cory Myers, Paul Fiore, and J.P. Letellier, “Rapid Development of Signal Processors and the RASSP Program,” Proceedings of the 1994 IEEE International Workshop on Rapid System Prototyping, June 1994.

[6] Fred Shirley, “Rapid Prototyping of Application-Specific Signal Processors Architectural Issues,” SPIE, July 1994.

[7] Sanders RASSP Team, “RASSP Process Document,, Sanders RASSP Team Document Number AVY-L-S-00078-101-A, June 1994.

[8] Sanders RASSP Team, “RASSP Architecture Metrics,” Sanders RASSP Team Document Number AVY-L-S-00076-101-A, June 1994.

[9] Sanders RASSP Team, “RASSP Architecture Issues,” Lockheed RASSP Team Document Number AVY-L-S-00080-101-H, June 1994.

[10] S. Famorzadeh, T.W. Egolf, V.K. Madisetti, “Rapid Systems Prototyping Models &Views,” Georgia Tech, June 10, 1994.

[11] Sanders RASSP Team, “RASSP Process Document,” AVY-L-S-00078-101-B.

[12] Jim Summers, Motorola, “Rapid Prototyping and the RASSP Design Environment (RDE)."

[13] “Reengineering and Beyond,” Boston Consulting Group, 1993.

[14] M. Hammer, J. Champy, “Reengineering the Corporation: a Manifesto for Business Revolution,” Harper Business, 1993.

[15] Hughes Aircraft Company Radar Systems, McDonnell Douglas Aerospace, “Advanced Design for Quality Avionic Systems.” March 1993.

[16] Advanced Avionics Architecture and Technology Review Final Report, Volume 1: Avionics Technology, 6 August 1993, Naval Air Systems Command, Arlington, VA.

[17] Architecture Specification for Pave Pillar Avionics, Air Force document number SPA90099001A, January 1987.

[18] JIAWG Advanced Avionics Architecture (A3) Standard, JIAWG document number J87-01, 1 December 1991.

[19] Dechant, Jagodnik and Wood, The Advanced Onboard Signal Processor: Brassboard Development and Space Qualification Plans, part of RADC-TR-87-268, January 1988, Rome Air Development Center, Griffis Air Force Base, NY.

[20] SAFENET Tutorial, NGCR Users Conference, 16-18 February 1993, Space and Naval Warfare Command, Arlington, VA.

[21] Swartzlander and Yang, AOSP Macro Function Signal Processor VHSIC Insertion, part of RADC-TR-87-268, January 1988, Rome Air Development Center, Griffis Air Force Base, NY.

[22] Department of Defense Trusted Computer Systems Evaluation Criteria (“Orange Book”), DOD 5200.28-STD, December 1985.

[23] Larry Scanlan, “The Road to 4X,” in the Proceedings of the Second Annual RASSP Conference, July 1995.

[24] B. Hood, C. Myers, “RASSP: Viewpoint from a Prime Developer,” Proceedings on the First Annual RASSP Conference, August 1994.

[25] Honeywell, “Graphics Processor Description VHDL Methodology: Working Document,” 1991.

[26] W. Lee, M. McCollough, M. Vahey, “Classification of VHDL Models,” Proceedings of the Spring 1995 VIUF Conference, April 1995.

[27] R. Dreiling, P. Kalutkiewicz, M. McCollough “Model Development within the RASSP Virtual Company Environment,” Proceedings of the Spring 1995 VIUF Conference, April 1995.

[28] Vahey et al, “A Virtual Prototype VHDL Development Methodology.” VIUF Spring 95.

[29] Dreiling et all, “Model Development within the RASSP Virtual Company Environment,” VIUF Spring 95.

[30] R. Dreiling, “Processes and Experiences in VHDL TopDown Design,” First Annual Conference on the Rapid Prototyping of Application Specific Signal Processors, August 15-18, 1994.

[31] S. Famorzadeh, R. Dreiling, M. Falco, et al., “Rapid Prototyping of DSP Algorithms on COTS Systems, " First Annual Conference on the Rapid Prototyping of Application Specific Signal Processors, August 15-18, 1994.

[32] “Hughes to gather IR data on theater missiles,” Aviation Week and Space Technology, vol. 140, num. 21, P 20,21, May 23, 1994.

[33] CUSeeMe Advanced technology Group in network resources, Cornell University, Information Technology Department.

Acknowledgment

The Lockheed RASSP team is under contract to the Naval Research Laboratory, 4555 Overlook Ave., SW, Washington, DC 20375-5326. The Sponsoring Agency is the Advanced Research Projects Agency, Electronic System Technology Office, 3701 North Fairfax Drive, Arlington, VA 22203-1714. The Lockheed RASSP team consists of Lockheed Sanders, Inc., Hughes, and ISX.

W. Hood
Lockheed Martin
PTP2-Coo, PO Box 868, 65 River Rd.
Nashua, NH 03061-0868
whood@rocket.sanders.com

Lockheed Martin Advanced Technologies Laboratories RASSP Second Year Overview

James E. Saultz

Abstract

The goal of the ARPA/Tri-Service-sponsored Rapid Prototyping of Application-Specific Signal Processors (RASSP) program is to improve by at least a factor of four the cost and time needed to develop and manufacture signal processors. The approach to reaching the program's goal is based on three technology thrusts; methodology, Model Year Architecture, and infrastructure (Enterprise). Using this triad of technology thrusts, Lockheed Martin Advanced Technology Laboratories' (ATL) RASSP team composed of an alliance of companies has implemented the first baseline RASSP system, which represents a significant advance over today's state-of-the-art. The methodology and tools have been used to demonstrate cost and design cycle improvements on the benchmark virtual prototype (VP) and have resulted in the development of a hardware/software system that demonstrated first-pass success. Additional developments being performed during the last two years of the program will provide further benefits, enabling demonstration of 4X improvements in cost and time-to-market. This paper provides an overview/update of the progress since the 1994 Annual Review.

1. Introduction

Low cost is becoming *the* key factor in enabling next-generation warfighting capabilities for the majority of emerging military systems. Many applications, such as the digitized battlefield, are characterized by the need for low-cost, high-throughput signal processing capabilities, which are often distributed in nature. The span of processing runs from simple speech processing to complex AAW Radar Systems for AEGIS. The 4X cost and cycle time improvements provided by RASSP will provide the ability to affordably apply state-of-the-art commercial and military-specific technology to make these capabilities a reality.

The Lockheed Martin ATL RASSP program approach to satisfying the RASSP goals is based on implementing the three technology thrusts: methodology, Model Year architecture, and infrastructure. The methodology is based on a concurrent/collaborative approach that embraces the full hierarchy, from requirements to manufacturing product data descriptions. The Model Year Architecture focuses on the leveraging of COTS technology, coupled with designing flexible, functional interfaces to enable regular, low-cost technology upgrades. The infrastructure (Enterprise) system enables the methodology and Model Year Architecture approach across multi-discipline, concurrent-engineering teams by providing integrated workflows, data, and network services. The resulting capability is a much greater capability than the sum of its parts, enabling the concurrent/collaborative virtual corporation of the future.

2. ATL Program Approach

The ATL RASSP team strategy for development and deployment of RASSP was to assemble a world-class team of leaders in all of the required technical disciplines. The team, shown in Figure 1, was chosen based on demonstrated technological leadership, significant ongoing investments in related areas and a vision of the future that aligned with the RASSP program goals. During the second year, all parts of the RASSP approach have been developed and integrated into a design environment that is supported by a well-defined methodology. The ATL RASSP team has continued to focus on a process and design environment that the user community can easily embrace.

At this milestone in the program we believe it would have been impossible to have made the advancements to date without leveraging the significant ongoing investments of our team members. While many of our team members are commercial vendors with their own sets of tools, the RASSP focus has been to develop standards-based processes, architectures and information standards to support reuse, design interchange, and the ability to customize instantiations of RASSP for individualized corporate use, as shown in Table 1. Because tools will have the shortest half-life on the program, one of RASSP's legacies will be the interoperability standards developed.

3. Year 2 Status

During the second year of the program, implementation of the approaches defined during the first year have begun in earnest. The developments to date have led to early commercialization of RASSP-based products, as shown in Table 2. These and other RASSP developments are demonstrating the benefits of virtual prototyping and hardware/software codesign on first-pass design success. Accomplishments that have shown particular promise include:

- Capture of the RASSP methodology into standard (IDEF) process language for import into the Enterprise System workflow management tools [1].
- Development of the Model Year Architecture reuse framework, which led to implementation of a Standard Virtual Interface (SVI) approach and supports low-cost retargeting and "plug and play" interconnect of processing nodes and interfaces. This approach is being proliferated to RASSP beta sites [2].
- Implementation of end-to-end virtual prototyping capability based on hardware/software codesign. Hardware/software codesign will be a significant factor in reducing integration and test, providing some of the largest contributors to achieving a 4x improvement [3].
- Development of advanced integration concepts that support multi-domain design verification across the design hierarchy.
- Development of a library-based, data-flow-graph-driven autocode process, which abstracts signal processing software generation and maintenance to the graph level [4].
- Development of the RASSP reuse classification hierarchy to accommodate a broader scope of reuse information

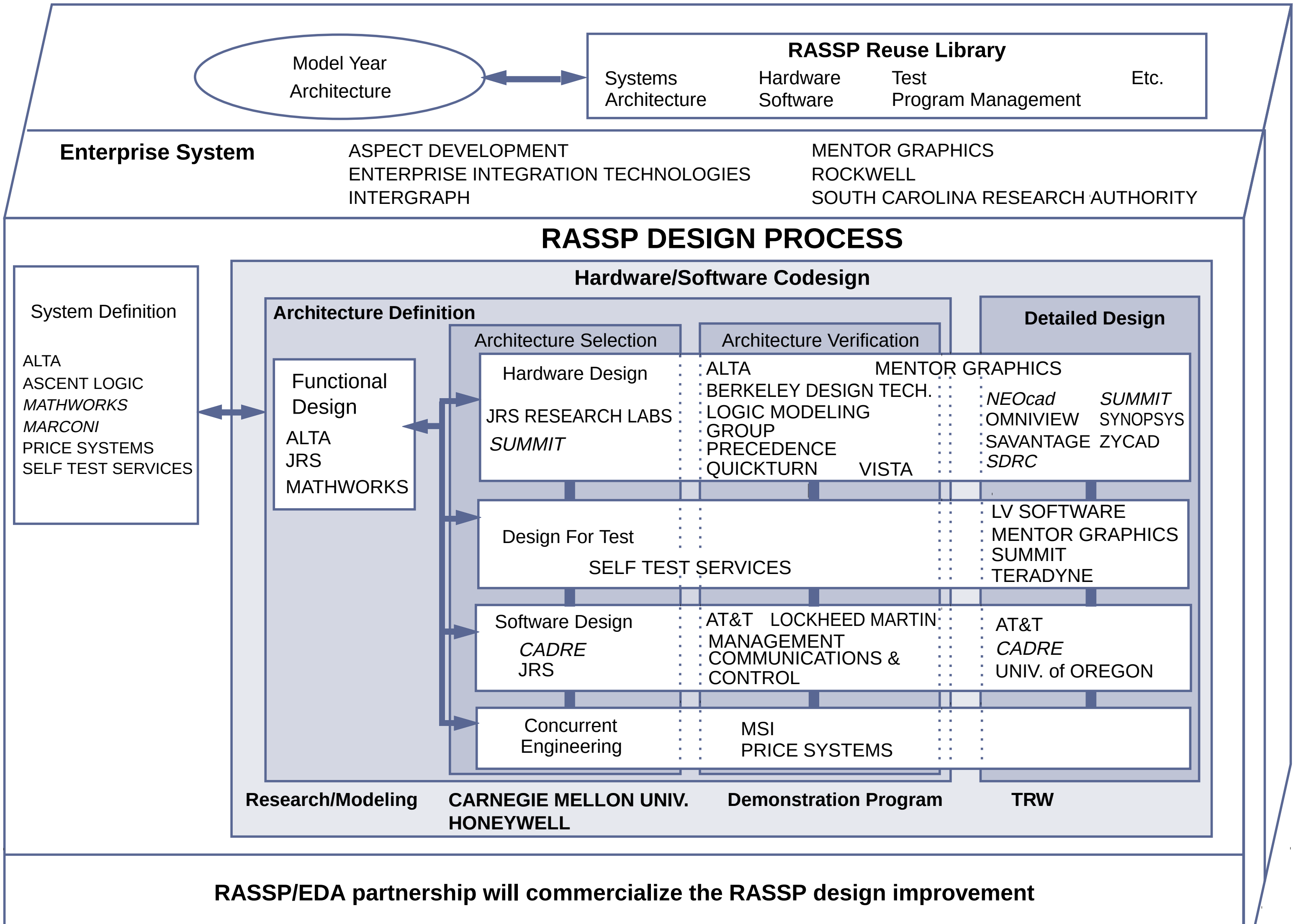


FIGURE 1. Lockheed Martin ATL RASSP Team

(such as VHDL models, algorithms, and software) beyond components. This classification scheme will be proposed as a standard to IEC and CFI.

- Use of the RASSP Information Model, based on STEP standards, as the basis for product data management. It is currently being examined as standard across Lockheed Martin [5].

Contributions of these developments toward the 4X goals of the program are further described in another RASSP paper being presented at the Second Annual Conference [6]. Additionally, proliferation of these early developments is ongoing. Installation/use of RASSP methodology and tools is already being planned/adopted by key Government and Industry organizations. This includes a number of internal Lockheed Martin sites and commercial companies (TRW, Allied Signal, Honeywell, Litton Data Systems, Rockwell, and Westinghouse). Several Army installations (ARL, Ft. Monmouth and NVL, Ft. Belvoir) also plan participation.

A summary of the more detailed developments is provided in the following sections for the major portions of the Lockheed Martin program.

3.1 Methodology

The RASSP Methodology has gone through a second year update. The design portion of the process is shown in Figure 2. During the year, the team has defined the top-level methodology and captured it in workflow processes using the Information Data Exchange Format (IDEF) on the Enterprise System's methodology manager.

The RASSP Design Process starts with the **System Definition** Phase. The primary functions being addressed in this portion of the design process are these:

1. System Requirements Analysis and Refinement,
2. Functional Analysis,
3. System Partitioning.

The system process captures customer requirements and converts these system-level needs into processing requirements, both functional and performance. The system process has no notion of

RASSP Technology Triad	Approach	Standard Information Models	Payoff
Methodology	• Concurrent/collaborative product development • Full product verification prior to manufacturing • Hardware/software codesign Reuse-based design	Workflows-IDEF	• Reduced time-to-market • First-pass success • Optimized solutions
Model Year Architecture	• Use architecture to standardize functional interfaces • Encapsulate elements into reuse library • Exploit COTS processing technology	Architecture/Hardware - VHDL, Software-Ada/C	• Increased design re-use • Reduced development cost • Low-cost upgrades - min. HW/SW breakage • New technology fielded sooner
Infrastructure (Enterprise)	• Use standards-based product data modeling/interchange • Exploit electronic highway • Use distributed data management	Product data flow - PDES/STEP, Express	• Seamless data transfer • Higher efficiency • Concurrent product development

Table 1: Standards-based RASSP information modeling.

Company	Tool	Capability	Date
Alta Group (Cadence)	MATLAB integration SPW/Bones integration (Ptolemy-based)	Algorithm import capability Flow graph/event simulator integration	3Q95 4Q95
Aspect Development	Aspect Explore system	Object-oriented class hierarchy	2Q95
Intergraph	Design Methodology Manager	Hierarchical workflow manager, project builder toolset	2Q95
JRS	NetSyn	Network synthesis	3Q95
Lockheed Martin	PRICE	UNIX-based cost estimation	4Q95
OmniView	FIDELITY	Board-level synthesis design advisor	Now
Precedence	SimMatrix	Simulation backplane extensions (VHDL, HSIM, Quickturn)	2Q-4Q/95

Table 2. RASSP is helping to drive commercial market, as shown in these commercialization announcements (as of June 1995).

either hardware versus software functionality or processor implementation.

The **Architecture Process** transforms processing requirements into candidate architectures composed of hardware and software elements, with support for codesign and *co-verification* at all steps. A conceptual view of the RASSP codesign and virtual prototyping approach is shown in Figure 3. The process truly embraces a hardware/software codesign methodology by taking functional processing requirements and iteratively allocating them to hardware/software in an experimentation/verification process. The process is inseparable from the software process, sharing in generation and verification of detailed code. The architecture process results in a detailed behavioral description of the processor hardware and definition of the software required for each processor in the system.

This part of the design is done by using a hardware/software (HW/SW) codesign approach, which refers to the simultaneous consideration of hardware and software within the design process [7]. The process begins with an architecture-independent data flow graph(s) representing the signal processing. During the process of selecting an architecture, the nodes of the data flow graph(s) are allocated to hardware or software. The graph nodes allocated to software are mapped to the multiple processors in the architecture, and performance estimates are generated based on timing information associated with the processing primitives from which the graphs are constructed. Alternative hardware architectures are developed and the system is optimized using execution times estimated for the target hardware. Functional simulation is used to verify that the generated code is consistent with the functional baseline. Performance simulation provides the next level of assurance that all throughput requirements are met by using lower level models, including the operating system, scheduling, and support software characteristics [8]. Finally, hierarchical architecture verification of the architecture is established using selective performance and functional simulation at the ISA and/or Register Transfer Level (RTL) level.

In the **Detailed Design** process, selective performance and full functional simulation are performed again. At this point, however, the design has progressed to the point where simulation at the RTL and logic levels is most appropriate. Verification of the designs at this level is necessary prior to release to manufacturing. It is important to note that pieces of the design may be in different stages of the overall process, based on the risk analysis performed in each development cycle. For example, if it is obvious to the designers during systems analysis that they will need a new custom hardware processor to meet the requirements, they may accelerate the design of the custom processor while the overall signal processor design is still in the architecture process.

3.2 Model Year Architecture

The Model Year Architecture task is the main contributor to the processor architecture portion of the RASSP technology triad. This task is providing a framework for developing application-specific signal processor architecture designs that encourages and facilitates hardware and software reuse, upgradability, and technology insertion. The Model Year Architecture framework, shown in Figure 4, is composed of the following elements:

- *Functional Architecture* -- A high-level hardware - architecture that provides a starting point for developing application-specific architectures within necessary constraints to ensure reuse, upgradability, and technology insertion.
- *Modular Software Architecture* -- A software architecture that is readily portable and upgradable because of its modularity, standardization on a required set of services. This architecture supports a new paradigm for application

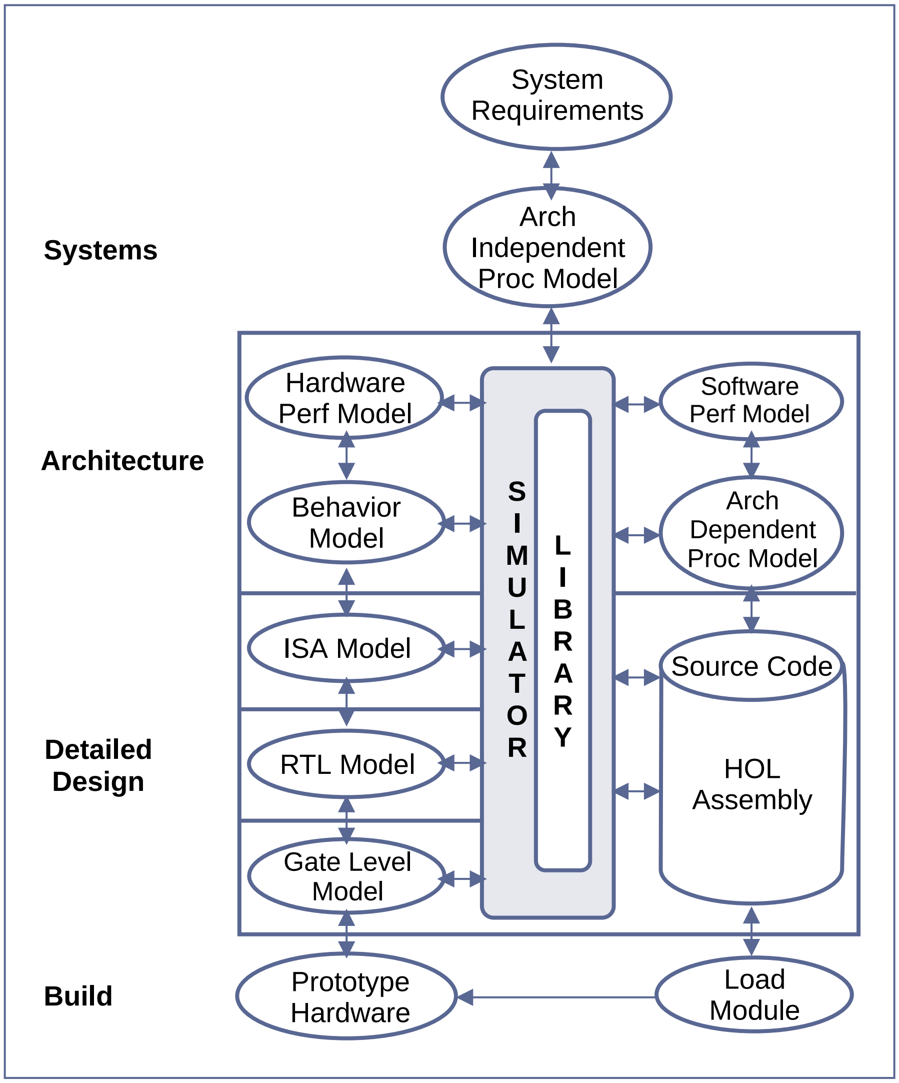


Figure 2. Hardware/software codesign view of the RASSP design process

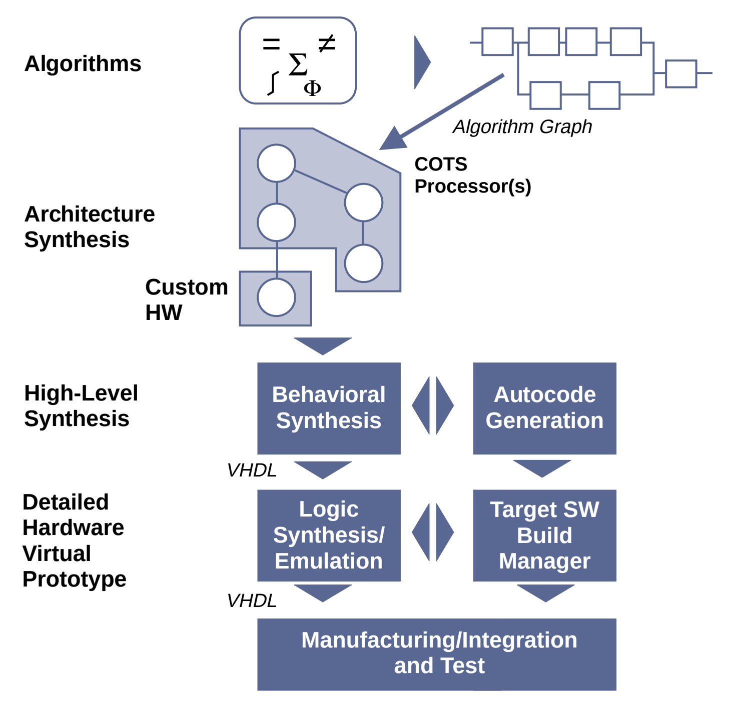


Figure 3. Codesign and virtual prototyping closes the technology gap.

software generation, which promotes automated re-generation of HOL code from tar get-independent primitives instead of porting existing HOL code for processor upgrades.

- *Encapsulated Library Components* -- Architectural-level reuse library elements that incorporate encapsulations or wrappers to implement a functional, technology-independent interface referred to as a Standard Virtual Interface (SVI). Such encapsulations support "plug-and-play" interoperability among library elements, which provides the benefits of decreased time to implement design upgrades and generate architectural library components.
- *Design Guidelines and Constraints* -- These provide required information to the designer for effective use of the Functional and Software Architectures general use of encapsulated libraries and the encapsulation procedures themselves.

During this year, the team has defined the key Model Year Architecture concepts of encapsulation, functional interfaces, and layering and is implementing and refining them through a number of ongoing experiments. The team is also finalizing baseline specifications, which include the SVI Specification, Hardware

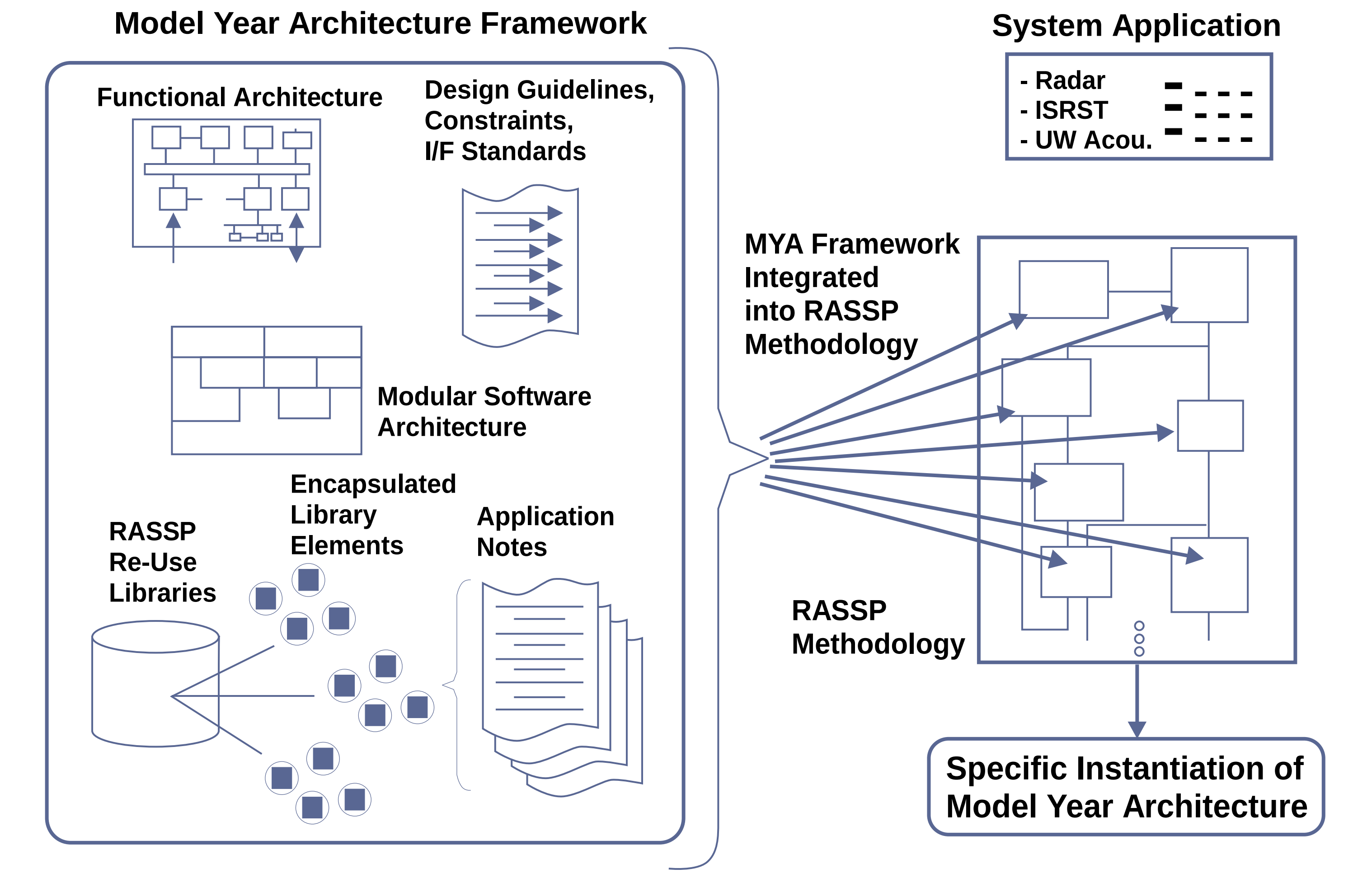

Model Year Architecture Framework
Functional Architecture
Design Guidelines, Constraints, I/F Standards
Modular Software Architecture
Encapsulated Library Elements
RASSP Re-Use Libraries
Application Notes
System Application
- Radar
- ISRST
- UW Acou.
MYA Framework Integrated into RASSP Methodology
RASSP Methodology
Specific Instantiation of Model Year Architecture

Figure 4. Model Year Architecture approach.

Architecture Specification, Software Architecture Element Specification, and the Model Year Architecture Reuse Element Specification. Additionally, implementation experiments, which involve writing VHDL encapsulations to implement the SVI for several processors and buses, have been performed.

Integration of the Model Year Architecture Framework into the Design Methodology and Enterprise Framework to achieve the required synergy between the elements of the technology triad is ongoing. The initial integration of the Model Year Architecture Framework based on these baseline specifications is expected by the end of 1995.

3.3 Design environment

The development of tools that support the RASSP methodology and Model Year Architecture have made major progress during the second year. The tool developments and integration support the full HW/SW codesign approach and are fundamental to achieving the RASSP 4X goals. The tool developments that have taken place in Year 2 are summarized in Table 3. A robust set of design and verification tools, which allow creating virtual prototypes that can easily be turned into producible products, are enablers for the methodology and Model Year Architecture parts of the triad. The following sections describe the design environment enhancements for the systems, architecture, software, and hardware elements.

Systems -- The system definition process is a front-end engineering task, where signal processing concepts are developed and top-level tradeoffs are performed to determine the signal processing subsystem requirements. As a part of the RASSP program, the team is performing integration of multi-discipline capabilities into true concurrent engineering environment. This environment consists of three major tools:

- Ascent Logic's RDD-100,
- Management Sciences' RAM/ILS toolset, and
- Lockheed Martin PRICE Systems' cost estimating tools. These tools are used for capturing and tracking system engineering requirements, describing the functional behavior of the signal processor, allocating the requirements to signal processing subsystems, performing high-level reliability and maintainability trade-off analyses, and performing parametric-based cost estimations for the signal processor's entire life cycle. The integration of these tools on the RASSP program will give the system engineer the capability to perform high-level trade-off analyses.

Architecture -- A set of tools to assist the designer in partitioning and mapping a functional application onto a potentially large number of computing nodes is a major requirement for meeting RASSP's time-to-market and reuse goals. JRS's NetSyn tool is the first available tool to assist in performing HW/SW codesign for architectural trade-offs. The RASSP team is currently integrating NetSyn with tools from other disciplines to enable designers to perform concurrent engineering trade-offs. The resulting RASSP capability will be the ability to efficiently evaluate a number of varying architecture approaches for a particular application and to generate top-level size, weight, cost, reliability, and performance estimates.

Once an architecture has been selected, more detailed verification of the implementation is required, which will likely be composed at any point in time of existing hardware and software elements, existing models, and new components. What is required at this level of the design hierarchy is a robust simulation capability that allows the designer to iteratively verify the design in a hierarchical manner, as shown in Figure 5. The RASSP program is making multi-domain simulation capabilities available through the productization of the Ptolemy-based Heterogeneous Simulation Interoperability Mechanism (HSIM) by Berkeley Design Technologies. Estimates indicate that the integration cost using HSIM was reduced by a factor of 8 over traditional integration approaches. Additional cross-domain integrations performed to date include integration of HSIM to the Precedence simulation backplane, as well as integration of VHDL and emulation (QUICKTURN) environments into the backplane. During Year 3, the team will develop a graphical user interface to support efficient hierarchical simulation partitioning, invocation, and visualization. The ATL RASSP team has demonstrated flexible application mapping and code verification on multiprocessor testbeds this year using Lockheed Martin's Graphical Entry Distributed Application Environment (GEDAE).

One of the design approaches that has developed over the past year is the use of VHDL to convey design information from the initial multiprocessor system concept through synthesizable chip descriptions. The team's efforts have focused on developing a VHDL Performance model interoperability standard and object-oriented extensions to support high-level modeling. This year, the team defined a Performance Model interoperability standard and developed an example (SAR benchmark) model using this approach. Models have been distributed to several organizations (TRW, Vista, HTC, Uva, JRS, and MIT) and have demonstrated more than 100X improvement in simulation time over traditional, ISA-level approaches. Honeywell is developing readily reconfigurable, generic libraries, with initial libraries already delivered, to support rapid trade-offs.

Object-oriented extensions to VHDL to support higher levels of abstraction in model definition and to support reuse have been developed. The approach taken has been to develop an object-oriented preprocessor that results in fully compliant IEEE 1076 VHDL code. VISTA has developed and demonstrated a prototype version of this preprocessor, which is now undergoing detailed evaluations.

Software -- One of the key developments on ATL's RASSP program is the development and implementation of a library-based, data-flow-graph-driven autocode process, which abstracts signal processing software generation and maintenance to the graph level. Data flow graphs represent the required signal processing using the Processing Graph Method (PGM). This representation is totally architecture-independent. Thus, as hardware is upgraded, the application description at the graph level remains constant.

During architecture selection, the processing represented by the nodes in the graph is allocated to hardware or software using

Table 3. RASSP methodology and Model Year Architecture productivity tool development.

Design Area	Company	Participation	Tools	2nd Year Enhancements
Architecture Definition	Alta Group of Cadence	Signal processing algorithm behavioral simulations and architectural simulation and performance verification	BONeS, SPW	Integrated SPW and BONeS using BDT HSIM. SPW/MATLAB interface allows MATLAB "M" files to be included in SPW as library block. SPW JRS Netsyn integration support multiprocessors designs.
System Definition	Ascent Logic Corporation	System requirements definition and functional decomposition	RDD-100	Integrating RDD-100 outputs with JRS Netsyn, MSI Rel/Maintainability and PRICE estimating tools allows concurrent high-level system tradeoffs.
Enterprise System	Aspect Development, Inc.	Design reuse library management and system component information	CLMS	Developed the RASSP Reuse Data Manager (RRDM) and integration into Enterprise System.
Architecture Verification	AT&T	Multiprocessor/Parallel processor software debugger	CDEM	Distributed debugger for COTS-based processors.
Architecture Verification	Berkeley Design Technology , Inc. (BDT)	Environment for simulation and prototyping of heterogeneous systems	HSIM (Ptolemy)	Productized Ptolemy kernel into HSIM enables cosimulation of heterogeneous high-level decision tools. Integration of HSIM to Precendence backplane.
Research	Carnegie Mellon University (CMU)	Architecture partitioning mapping tools		Developing processor architecture mapping schemes.
Enterprise System	Enterprise Integration Technologies	Networking services		Developing secure networking capability to support virtual corporations, exploit electronic commerce.
Modeling	Honeywell Technology Center (HTC)	VHDL Performance Modeling		Developing a generic parametric library of VHDL performance models and interoperability guidelines.
Enterprise System	Intergraph Corporation	Enterprise framework - product data and workflow management	DMM, DM2.0	Developed enterprise-level object-oriented data management to support RASSP design reuse concepts.
Architecture Selection	JRS Research Laboratories, Inc. (JRS)	Integrated architecture tradeoff and synthesis	IDAS	Demonstrate the viability of Network Synthesis System (Netsyn) for rapid architecture tradeoffs. Architectural selection toolset: Netsyn, RDD-100, MSI Rel/Maintainability & PRICE cost analysis.
Detailed Design	LogicVision Software, Inc. (LV SW)	Electronics systems test automation tools for insertion, synthesis and fault grading of BIST structures	ICBIST	Developing a hierarchical integrated BIST methodology and tools for PCB, MCM and system-level test. Development is being integrated with LM ATL design approach.

Table 3. RASSP methodology and Model Year Architecture productivity tool development (cont.)

Design Area	Company	Participation	Tools	2nd Year Enhancements
Architecture/ Software	Management Communications & Control, Inc. (MCCI)	Development of multiprocessor DSP autocode tools and distributed run-time scheduling and control	GrTT, uPIDgen	Developing and integrating suite of tools to support the automatic code generation for COTS processors from signal flow graphs (PGM)
System & Distributed Design	Management Sciences, Inc. (MSI)	Reliability, availability, and maintainability analysis	RAM/ILS	Developing and integrating suite of tools for reliability/maintainability predictions, FMECA and production assessment.
Detailed Design	Mentor Graphics Corporation (MGC)	Integration services and HW design of component-based automation tools	Falcon, QuickVHDL, etc.	Supporting integration of Enterprise System and design tools that to build COTS-based multiprocessors.
Detailed Design	Omniview , Inc.	Tool extension to synthesize DSP boards and parallel/distributed systems	Fidelity	Extending a general purpose board synthesis capability for signal processing boards.
Arch/Design Verifications	Precedence, Inc.	Simulation backplane allows multiple simulations to run concurrently	SimMatrix	Extending SimMatrix capability to permit cosimulation of all levels of signal processor design.
System Definition	PRICE Systems	Parametric cost estimation	PRICE S, H, M	UNIX-based parametric cost estimating tools for doing HW, HW life cycle & SW.
Arch/Design Verification	Quickturn Systems	Integration of emulation & design tools	ASIC Emulator	Integration of emulation capability to design environment.
Enterprise System	Rockwell Aerospace	Workflow and information model development		Neutral workflow format process modeling language and enterprise model repository.
Enterprise System	SCRA	Electronic interface to manufacturing facilities		Standards-based manufacturing interface to support virtual prototyping between design and manufacturing organizations.
Total Test Methodology	Self Test Services (STS)	Design-for-test (DFT) methodology and tools		DFT methodology to support total design hierarchy.
Detailed Design	Synopsys, Inc.	VHDL source compilation and logic synthesis	VHDL/Design Compiler, Design Ware	Supporting use of synthesis tools for designing processors.
Demos	TRW	Demonstration of RASSP for ICNI application		Demonstrations of the RASSP concepts for signal processor design.
Software Debugging	University of Oregon	Software analysis and binary-to-binary translation of real-time software	TIBBIT , PIE	Performance Instrumentation Environment (PIE) for performance debugging.
Research Modeling	Vista Technologies, Inc.	Object-oriented VHDL extensions	OO-VHDL, StateVision	Developing OO-VHDL preprocessor to generate models and test benches.

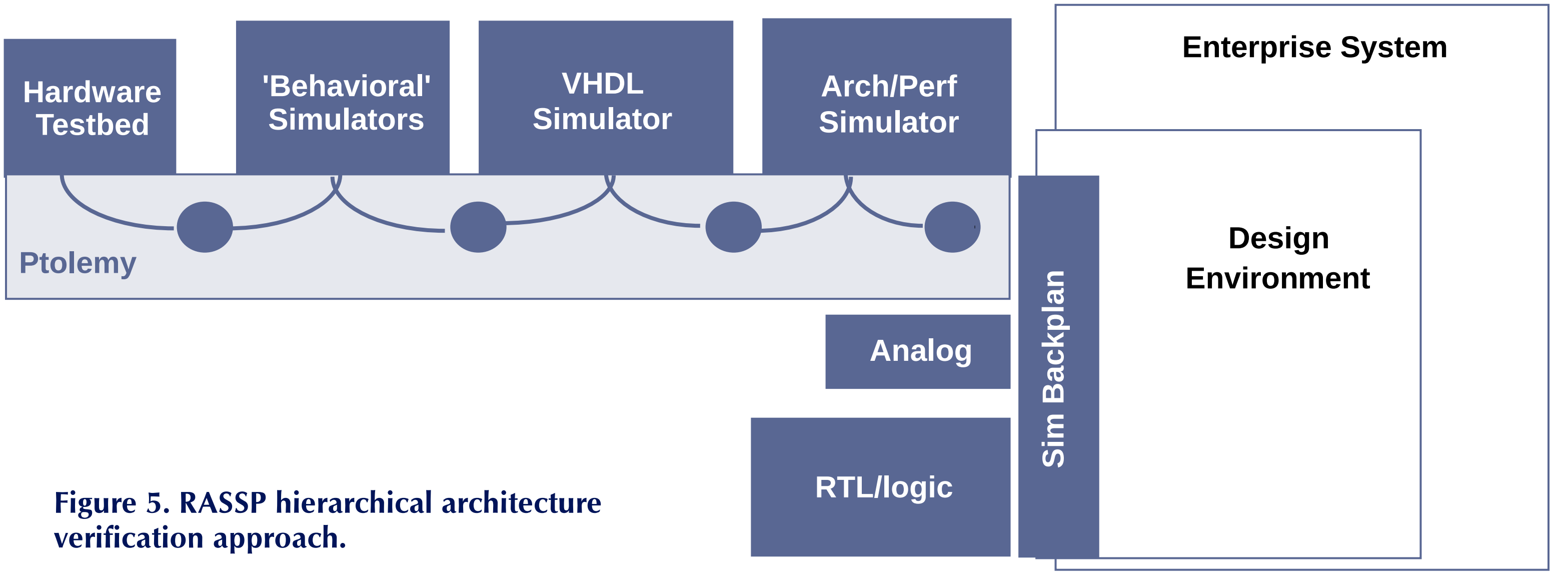


Figure 5. RASSP hierarchical architecture verification approach.

various programmable processors in the architecture. The description of the architecture, the mapping information, and the PGM graph are seamlessly passed from NetSyn to the Autocode tools. Connected groups of primitives assigned to the same processor represent graph partitions, which are automatically translated (using the MCCI-developed Autocode tools) to source code for the processor type to which the partition has been mapped. This translation uses the optimized math and signal processing libraries for the specified target processor. Along with the Autocode tools, MCCI has completed the design and is currently implementing a Run Time System to support the management and execution of the autocoded graphs on the target hardware. The Run Time System is being built with an open interface to operating system microkernels to facilitate porting to commercial products. The first integrated version of both the Autocode tools and the Run Time System will be released in 4Q95 and will support the MCOS operating system and Mercury signal processing application library (SAL). The Autocode tools and Run Time System will provide the recently initiated AN/UYS-2A upgrade program with both the ability to easily retarget PGM software to the new hardware and the ability to upgrade the hardware without modifying the software at the graph level. This effort will represent the first real application of automated code generation and run-time support targeted to commercial processors.

Hardware/Design for Test (DFT) -- Integration of DFT activities are focused on knitting together chip to system testability over the entire product life cycle from design verification and manufacturing acceptance through field support. The DFT methodology contributes to achievement of the overall RASSP goals in two ways. First, adoption of DFT practices, such as being developed and practiced within industry , results in reduced cycle time, reduced cost, improved quality, predictable schedules (including integration and test), improved time-to-market, and most importantly time-to-profit. Secondly, the structured DFT methodology provides improvement of the DFT process itself compared to current industry practice. This is achieved by the introduction of proven system engineering practices, such as the consolidation of test requirements. It is also achieved by leveraging top-down development of the overall RASSP methodology to flow-down the test strategies and architecture from the system to chip packaging levels and across life-cycle phases of the product.

The team has captured the blueprints for the process enhancements in the recently completed DFT Methodology and Testability Architecture documents. Additionally, activities to implement the process enhancements are proceeding.

3.4 Enterprise System

The RASSP Enterprise System development during Year 2 has concentrated on developing the key elements required to support the RASSP Methodology and Model Year Architecture concepts. The conceptual view of the RASSP Enterprise System is shown in Figure 6. The Enterprise System development has been broken into the major areas described below.

The **Enterprise System integration effort** has made significant progress and has resulted in a concept of operations, which has been used to drive the vision of our detailed developments. The team has completed enterprise-level integration of several of the elements of the methodology, which includes implementation of the workflows into the Intergraph Design Methodology Manager integration of the tools associated with those workflows into the Desktop manager and integration of the tool's data into the RASSP Product Data Management System. The methodology/workflows for the complete hierarchy of design steps will be completed early in Year 3.

The **Enterprise Data Management** portion of the Enterprise System was updated with the new Intergraph DM2.0 Data Management tool (Metaphase Data Management tool). DM2.0 is the central part of the RASSP Product Data Management System, and is an Object-Oriented Data Management tool that supports the RASSP Configuration Management and Authorization Models and policies that were developed this year. The use of DM2.0 ensures that RASSP is supported by a viable commercial approach that will be distributed by a commercial supplier.

Aspect and Lockheed-Martin are developing the RASSP **Reuse Design Object Classification Hierarchy** for hardware, software, architecture, systems, VHDL model, and algorithm design objects. Aspect Development is working closely with ATL to standardize this hierarchy as a product of the RASSP program. The specific Reuse Data Management approach being pursued takes full advantage of Aspect's Object-Oriented Component Library System and the latest reuse tool, referred to as Explore-CIS. An example browser window is shown in Figure 7.

The RASSP Enterprise System is addressing the **Design-to-Manufacturing Interface** requirements. During the second year, the team has implemented an approach for interfacing the RASSP board design tool outputs through a STEP AP210 approach. The Manufacturing Interface portion of the development is also leveraging the SCRA PreAmp development toolset developed under a NIST program. The RASSP Design-to-Manufacturing approach is being implemented at the Lockheed Martin Ocala Manufacturing Facility in Florida. The design-to-manufacturing effort is looking at building a bridge between STEP and EDIF.

RASSP funding also supports extensive **design tool and data integration**. Tool integrations, which enable bi-directional data exchange and synchronization through the graphical user interface and in batch modes, include:

1. Intergraph Corporation's Design Methodology Manager (DMM),
2. Intergraph's new Data Manager (DM2.0), which will manage product data in the RASSP enterprise environment,
3. Mentor Graphics' Library Management System (LMS) and Design Architect.

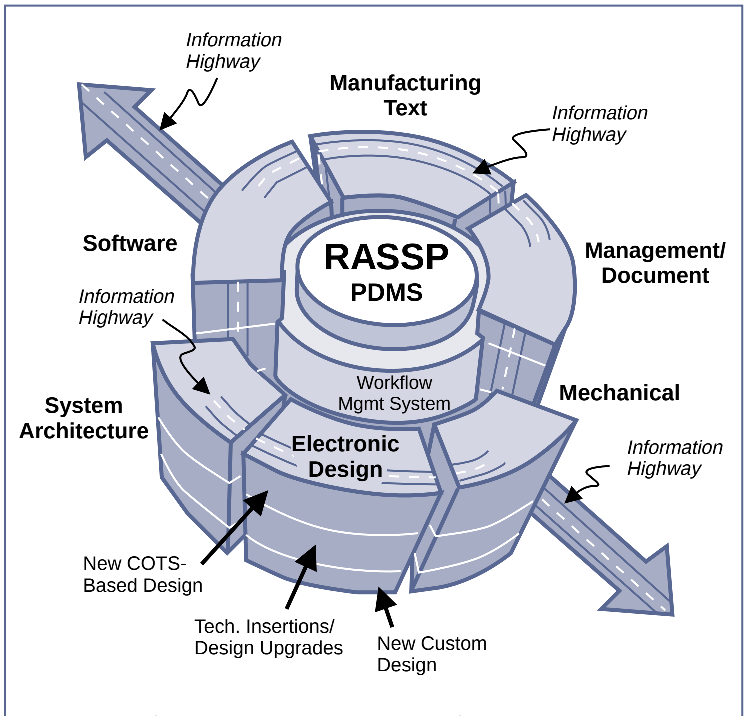
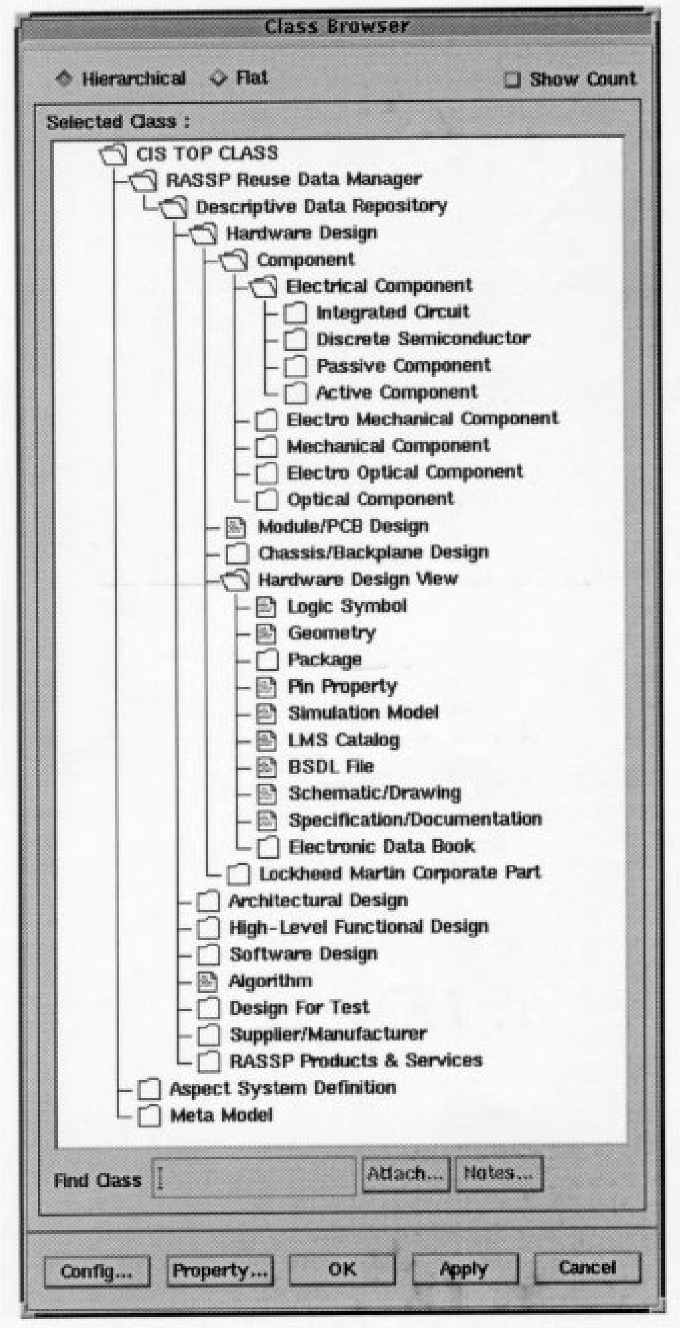


Figure 6. RASSP Enterprise System.



- CIS TOP CLASS
  - RASSP Reuse Data Manager
    - Descriptive Data Repository
      - Hardware Design
        - Component
          - Electrical Component
            - Integrated Circuit
            - Discrete Semiconductor
            - Passive Component
            - Active Component
          - Electro Mechanical Component
          - Mechanical Component
          - Electro Optical Component
          - Optical Component
        - Module/PCB Design
        - Chassis/Backplane Design
        - Hardware Design View
          - Logic Symbol
          - Geometry
          - Package
          - Pin Property
          - Simulation Model
          - LMS Catalog
          - BSDL File
          - Schematic/Drawing
          - Specification/Documentation
          - Electronic Data Book
        - Lockheed Martin Corporate Part
      - Architectural Design
      - High-Level Functional Design
      - Software Design
      - Algorithm
      - Design For Test
      - Supplier/Manufacturer
      - RASSP Products & Services
  - Aspect System Definition
  - Meta Model

Figure 7. The RASSP Reuse Design Object Classification Hierarchy in the Explore-CIS class browser window.

In the **Collaboration Design area**, several collaborative tools were identified and will lead to a selection and integration during Year 3. The leading candidates are the ARPA-sponsored MECE tool developed by the Lockheed Martin Palo Alto Laboratories and the SRI Collaborative tool.

The final area being addressed in the Enterprise System is the **Network Services**. The ATL RASSP team member responsible for supplying Enterprise System Network Services is Enterprise Integration Technologies (EIT). The EIT group will be making a secure network server available for supporting the Design-to-Manufacturing Interface Experiment being run at the Lockheed Martin Ocala PWA shop. During the third year, the number of

RASSP sites using the EIT capability will be expanded to allow demonstrating the distributed design and manufacturing approach that is being developed. This capability will be used to demonstrate the virtual corporation being developed for the RASSP program.

3.5 Benchmarks, demonstrations, proliferation

A number of benchmark and demonstration efforts are ongoing to impact high-visibility military platforms, demonstrate RASSP technologies, and provide feedback for optimization. The largest activity is the AN/UYS-2A upgrade, which is currently beginning. This demonstration will provide greater than 30X capability upgrade to the UYS-2A processor aboard the LAMPS helicopter to support shallow water target detection/classification in first-quarter-1997 flight test. Major features include implementation of a 4 GFLOP Floating Point Commercial Arithmetic Processor (FCAP) SEM-E board in the AN/UYS2A, and use of the RASSP Autocode capability to enable cost-effective retargeting across a wide range of Navy programs.

TRW is using the RASSP system to develop a Spread-Spectrum Preprocessor (SSPP) targeted for the requirements of the Integrated Sensor System (ISS) and Joint Advanced Strike Technology (JAST) programs [9]. Spread-spectrum processing is a key part of many communication links, including JTIDS, EPLRS, GPS, and WNA. The JAST mission is to create affordable strike warfare systems. Several of JAST's methods to achieve its goals are parallel to RASSP, such as commonality, improved practices, reduced upgrade costs, and simulation. Specific cost savings goals are 33% O&S, 64% production, and 6% R&D. TRW has developed one virtual prototype SSPP optimized for maximum COTS usage. Three more virtual prototypes are planned for this year. In 1996, TRW will add more features and build one of the virtual prototypes in end-item hardware that will be available to the ISS and JSAT program for flight tests.

The KINDLING Demonstration is a small effort to apply portions of the RASSP process to a classified application. Use of graph-based autocode generation capabilities has demonstrated almost 10X improvement in algorithm software generation for pieces of this application. The team is also working on the definition of an Executable Specification, which is being considered as a potential Government procurement approach for upcoming programs.

The use of the RASSP methodology and design tools were used to design the Benchmark 1 Virtual Prototype [10]. The Benchmark 1 experience has lead to demonstrating an approach for applying VHDL to model full computing systems that contain upwards of hundreds of processor elements. A central theme is the promotion of true hardware/software codesign through the independent specification of the software and the hardware to support the rapid exploration of various software applications and mappings on many architectural candidates. Reduction of the design cycle-time to less than a half-hour for relatively complex applications such as the Synthetic Aperture Radar (SAR) allows multiple design iterations per day .

The team completed successful architecture verification and optimization of a full SAR DSP system containing 24 cooperating processor nodes running multiple iterations of the SAR application algorithm, which span several seconds of simulated real time. The system models provide early design verification via data-flow-graph-driven simulation of software as partitioned, mapped, and executed on the hardware architecture. This verified the RASSP virtual prototyping approach by providing early design verification of software as partitioned, mapped, and executed on the hardware architecture, prior to hardware manufacture. The benchmark also demonstrated hierarchical simulation and testbench data from virtual prototype to verify RTL-level descriptions of custom boards and FPGAs.

The team also implemented the RASSP methodology and demonstrated an object-oriented approach to autocode generation of command program for SAR. Benchmark 2 demonstration will show ease in retargeting the application for a Model Year product upgrade (i860 transition to 21060).

4 Summary

The Lockheed Martin ATL RASSP program progress during Year 2 has shown that many of the proposed concepts are achievable. These developments are being used to demonstrate the benefits in schedule, cost, and quality that can be achieved in signal processor design. The Benchmark 1 and 2 efforts have clearly demonstrated the adaptability and flexibility that can be achieved by using the RASSP methodology and design environment. The team's Benchmark 1 virtual prototype was focused on building a SAR processor using a COTS processor-based approach. Delay in delivery of the full functional/performance vendor part forced a mid-stream change in the COTS approach. Because the design had been captured in hierarchical models/simulations, it was very easy to retarget to an available COTS processor. This unplanned programmatic deviation is not uncommon among real-world programs.

The demonstrated capability to develop the first model year release of the system with a small perturbation in time and cost has convinced the ATL RASSP team that the virtual prototyping paradigm being pursued has fully proven its benefits. A minimum of 2X improvement in the schedule and cost, while maintaining the quality of the design, was demonstrated. The ATL Benchmark team plans to implement the Model Year 2 approach using an improved processor that will substantiate the Model Year Architecture methodology being pursued. The goal is to then show that a new set of algorithms (non-SAR application) could be easily mapped to the COTS-based architecture.

The UYS-2 program upgrades will demonstrate the benefits of the RASSP Methodology and design tools. The use of the virtual prototyping tools and automatic code generation tools coupled with new concepts, such as the ARPA Myrinet development, will demonstrate the ease of building COTS-based processors for multiple service applications with the RASSP concepts at significantly reduced cost and schedule.

In addition to demonstrating the use of RASSP concepts, the team will deploy a large number of the tools for use by the total international electronic design community. This is an exciting part of the program because it will help fund continued improvements to the tools and will allow the community to develop models and examples that can be used by the user community.

The RASSP concepts are being applied to other ARPA-Tri-Service programs. The first such program is the Afford-able Multi-Mission Manufacturing (AM3) program. The ATL RASSP team believes the RASSP enterprise concept and implementation will also have applications to many other design and manufacturing requirements.

References

[1] Bard, A., Finnie, E., Forte, M., Selvidge, W., Stavash, J., Tuck, M.C., and Wedgwood, J., *Workflow Modeling for Implementing Complex, CAD-Based, Design Methodologies*, Second Annual RASSP Conference Proceedings, 1995.

[2] Caracciolo, G. and Pridmore, J., *Reuse-Oriented Model Year Architectures for Rapid Prototyping*, Second Annual RASSP Conference Proceedings, 1995.

[3] Bard, A. and Schaming, W.B., *Hardware/Software Codesign in the Lockheed Martin Advanced Technology Laboratories' RASSP Program*, Second Annual RASSP Conference Proceedings, 1995.

[4] Robbins, Christopher, *Autocoding in the Lockheed Martin ATL RASSP Hardware/Software Codesign Process*, Second Annual RASSP Conference Proceedings, 1995.

[5] Bard, A., Chadha, B., Finnie, E., Kalathil, B., Selvidge, W., Tuck, M.C., and Welsh, J., *Integrated Process Control and Data Management in RASSP Enterprise Systems*, Second Annual RASSP Conference Proceedings, 1995.

[6] Pridmore, J., *Advanced Technology Laboratories' Path to 4X Improvements*, Second Annual RASSP Conference Proceedings, 1995.

[7] Bard, A., Myers, C., and Schaming, W.B., *Hardware/ Software Codesign*, RASSP Working Document, 1994.

[8] Hein, C. and Nasoff, D., *VHDL-Based Performance Modeling and Virtual Prototyping*, Second Annual RASSP Conference Proceedings, 1995 .

[9] Kuttner, C., *TRW RASSP Model Year 1 Spread Spectrum Pre-Processor*, Second Annual RASSP Conference Proceedings, 1995.

[10] Jaffe, R., Kline, W., and Pridgen, J., *RASSP Technology Insertion into the Synthetic Aperture Radar Image Processor Application*, Second Annual RASSP Conference Proceedings, 1995.

Jim Saultz
Martin Marietta
1 Federal Str. A&E, 2-W
Camden, NJ 08102
jsaultz@atl.ge.com

RASSP Steering Committee

ARPA (ETO)	
• Randy Harr	**Program Manager**
ARMY	
• Randy Reitmeyer	**Administrative COTR, Lockheed/Advanced Technology Labs**
• Arne Bard	**Technical COTR, Lockheed/Advanced Technology Labs**
NAVY	
• Ingham Mack (ONR)	
• Gerry Borsuk (ONR)	
• Joe Killiany (NRL)	**Administrative COTR, Lockheed/Sanders**
• J. P. Letellier (NRL)	**Technical COTR, Lockheed/Sanders**
AIR FORCE	
• Stan Wagner	**Educator Facilitator and Technology Base**
• John Hines	**COTRs**

RASSP Digest-Rapid Prototyping of Application Specific Signal Processors

The RASSP Digest is published quarterly and provides information for and about the RASSP Program and rapid systems development. For more information, contact Dr. Anthony Gadient or Dr. Vijay Madisetti, Editors, at the addresses below:

Dr. Anthony J. Gadient
Phone: 803-760-4082
FAX: 803-760-3349
Email: gadient@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

Dr. Vijay K. Madisetti
Phone: 404-853-9830
FAX: 404-853-9171
Email: vkm@ee.gatech.edu
Georgia Tech
Sch. of Elec. & Computer Eng.
Atlanta, GA 30332-0250

Kristi Adams
Managing Editor
Phone 803-760-3376
Email: adamsk@scra.org

Methodology

RASSP
Reinventing
Electronic
Design

Architecture Infrastructure

ARPA • Tri-Service

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • MMG • ADL

Calendar of Events

VHDL International Users Forum (VIUF) For More Information: Pam Rissman 415-329-0578 or fax: 415-324-3150	October 15-18, 1995	Boston, MA
GOMAC, RASSP Seminar For More Information: Ed Hakim 908-427-2185	October 18, 1995	Orlando, FL
CALS Expo For More Information: Dr. D. Brent Pope 202-775-1440 or brentpope@delphi.com	October 23-26, 1995	Long Beach, CA
DSP World Expo For More Information: Ann Harris 617-891-6000 or DSPWorld@world.std.com	October 24-26, 1995	Boston, MA
4th International HW/SW Co-Design Workshop For More Information: Prof. Don E. Thomas 412-268-3545 or thomas@ece.cmu.edu	March 18-20, 1996	Pittsburg, PA
NSF Workshop on Workflow and Process Automation in Information Systems For More Information: Prof. Amit Sheth 706-542-2310 or amit@cs.uga.edu	May 8-10, 1996	Athens, GA
Design Automation Conference (DAC) For More Information: MP Associates 303-530-4333	June 3-7, 1996	Las Vegas, NV

SCRA
5300 International Blvd.
N. Charleston, SC 29418

RASSP Digest

I
1996

RASSP - Rapid Prototyping of Application Specific Signal Processors

Vol. 3, 1st. Qtr. 1996

The Road to Enterprise Integration

Methodology
RASSP
Reinventing
Electronic
Design
Architecture Infrastructure
ARPA • Tri-Service

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • ADL

In This Issue

▸ Editor's Corner

▸ Prime Development

▸ Technology Base

RASSP Digest Theme: The Road to Enterprise Integration

Anthony J. Gadient and Vijay K. Madisetti

As organizations focus on core competencies, the need to bring diverse organizations together to satisfy the various resource needs required for large product developments (such as a new weapon system or automobile) is becoming increasingly important. Successfully forming a virtual corporation in a timely fashion faces several challenges. Flexible enterprise integration systems can provide the infrastructure needed to overcome these challenges.

This edition of the *The RASSP Digest* is focused upon the RASSP efforts to develop the enterprise integration infrastructure necessary to support collaborative design in a distributed, heterogeneous environment. This infrastructure is vital to enable the rapid formation of virtual organizations so important to business and the Department of Defense (DoD) today.

To achieve the enterprise integration objective of electronically enabling virtual collocation in time and space, four sets of related problems must be addressed.

- Connectivity
- Interoperability
- Security
- Business/Cultural

Connectivity implies that an organization be accessible via the information highway. Interoperability requires the ability for organizations using different applications on different platforms to be able to effectively and efficiently exchange information. This spans the range from different organizations using different word processors on different platforms (e.g., PC's and Macintosh's) being able to exchange information and make effective use of it, to the ability of different designers to share complex models of hardware and make effective use of these models. Security issues span the realm from encryption to support information exchange over public networks to authentication mechanisms that allow one to ensure they are communicating with whom they think.

By solving these three sets of related problems, new capabilities are enabled that represent services that an enterprise integration framework can provide. These services include capabilities such as workflow management systems, enterprise product data management systems either stand-alone or integrated with a workflow management system, and enterprise library management systems.

Lastly, there are numerous business and cultural issues that must be addressed to allow the effective application of enterprise integration technologies like those being developed on RASSP. The RASSP Education & Facilitation program is working to overcome these business and cultural barriers.

To cover the enterprise integration area, a collection of five papers has been assembled. The first two papers present the efforts of the RASSP prime contractors. The first paper, *Integrated Process Control and Data Management in RASSP Enterprise Systems*, by John Welsh, et. al. of the Lockheed Martin Advanced Technology Laboratories' (LM-ATL) RASSP team, describes the LM-ATL enterprise system, highlighting the components that make up that system and presenting a strategy by which the services provided by this enterprise framework can improve efficiency in task execution and information management. The second paper, *Enterprise Integration*, by James Chieks of the Lockheed Sanders RASSP team, highlights many of the technical and business/cultural difficulties involved in achieving enterprise integration. These two papers by the RASSP primes are followed by two invited papers which help to present a comprehensive view of the enterprise integration area. The first of these invited papers, *The National Industrial Information Infrastructure Protocols (NIIIP) Project*, by Richard Bolton of the NIIIP Project, presents an overview of this important, ARPA funded project. The NIIIP technical vision is to define ways for existing applications to inter-operate and to make the technologies fit together in a useful manner based on existing, emerging, and defacto standards such as ISO 10303 (STEP). The next invited paper, *Concurrent Engineering Wheels*, by Biren Prasad, Managing Editor of the *Concurrent Engineering Research and Applications Journal*, focuses on the topic of cooperative product development or concurrent engineering. Enabling concurrent engineering is one of the primary benefits provided by enterprise integration. The final paper, *Agility through Information Sharing: Results Achieved in a Production Environment*, presents the RASSP program's efforts to develop an agile manufacturing interface utilizing the RASSP enterprise integration capabilities. The results presented in this paper, a 10x reduction in design to manufacturing cycle-time and more than an 80% reduction in rework, highlight the benefits that can be achieved from enterprise integration.

Once again, *bon appetit*, for there is a lot to *Digest* in this important issue.

Anthony J. Gadient
SCRA
5300 International Blvd.
N. Charleston, SC 29418
gadient@scra.org

Vijay K. Madisetti
ECE,
Georgia Tech.
Atlanta, GA 30332-0250
vkm@ee.gatech.edu

Integrated Process Control and Data Management in RASSP Enterprise Systems

John Welsh, Biju Kalathil, Bipin Chadha, Mary Catherine Tuck, William Selvidge, Elisa Finnie, and Arne Bard

Abstract

The RASSP Enterprise System provides key automation support for multidisciplinary teams of engineers and managers in the execution of complex development projects. As a result, the system facilitates greatly improved productivity, as well as efficient program control and orderly management of design configurations. Core concepts of the RASSP Enterprise System include integration of tools and tool frameworks into an enterprise environment; program execution control through workflows; integrated data management functions; concurrent engineering team support; and integration of design engineering and manufacturing. This paper presents a strategy for the use of the RASSP environment, methodology/workflows, and information models to improve efficiency in task execution and information management on signal processor development projects.

1. Enterprise System Overview

The RASSP Enterprise System architecture is hierarchical, integrating individual design tools, as well as collections of tools, which are themselves integrated into specialized frameworks. This architecture includes provisions for purchasing systems and product data management systems [1]. The architecture also provides a distributed reuse system with an object-oriented repository at the enterprise level and coordinated local framework/tool libraries.

The concept of operation for the enterprise framework includes the ability to execute project plans, expressed as workflows, by teams of engineers. Execution of a workflow by a member of a design team, as indicated in Figure 1, initiates control commands to a CAD/CAE tool as relevant for the particular workflow step. This execution also initiates data transactions with the enterprise product data management system; local data management systems; and library systems, as relevant for the particular workflow step. In addition, project management tools are coupled with the enterprise environment, which receives regular status updates as workflow steps are executed. This process facilitates effective, non-interfering project management.

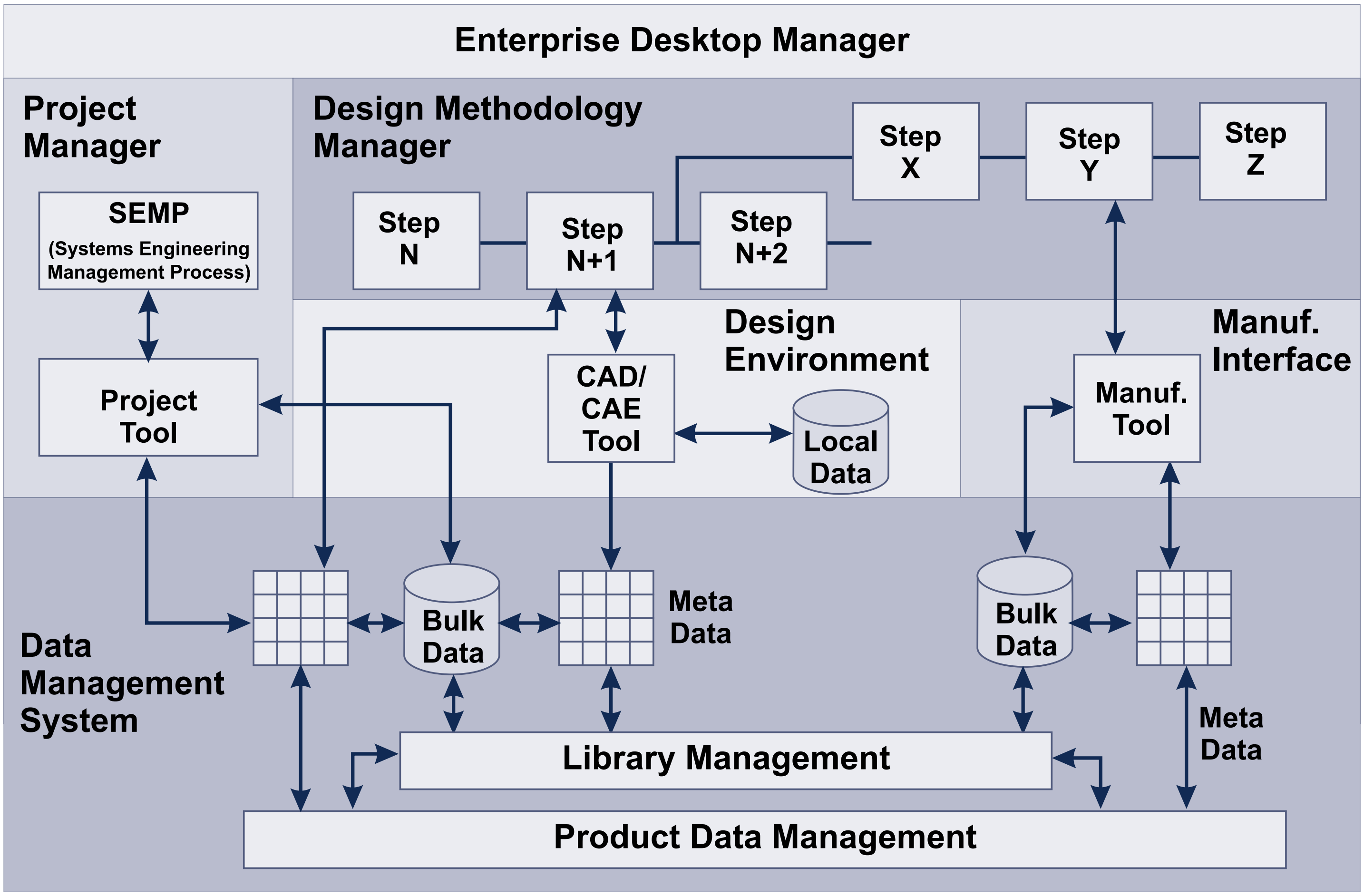


FIGURE 1. Enterprise System Organization

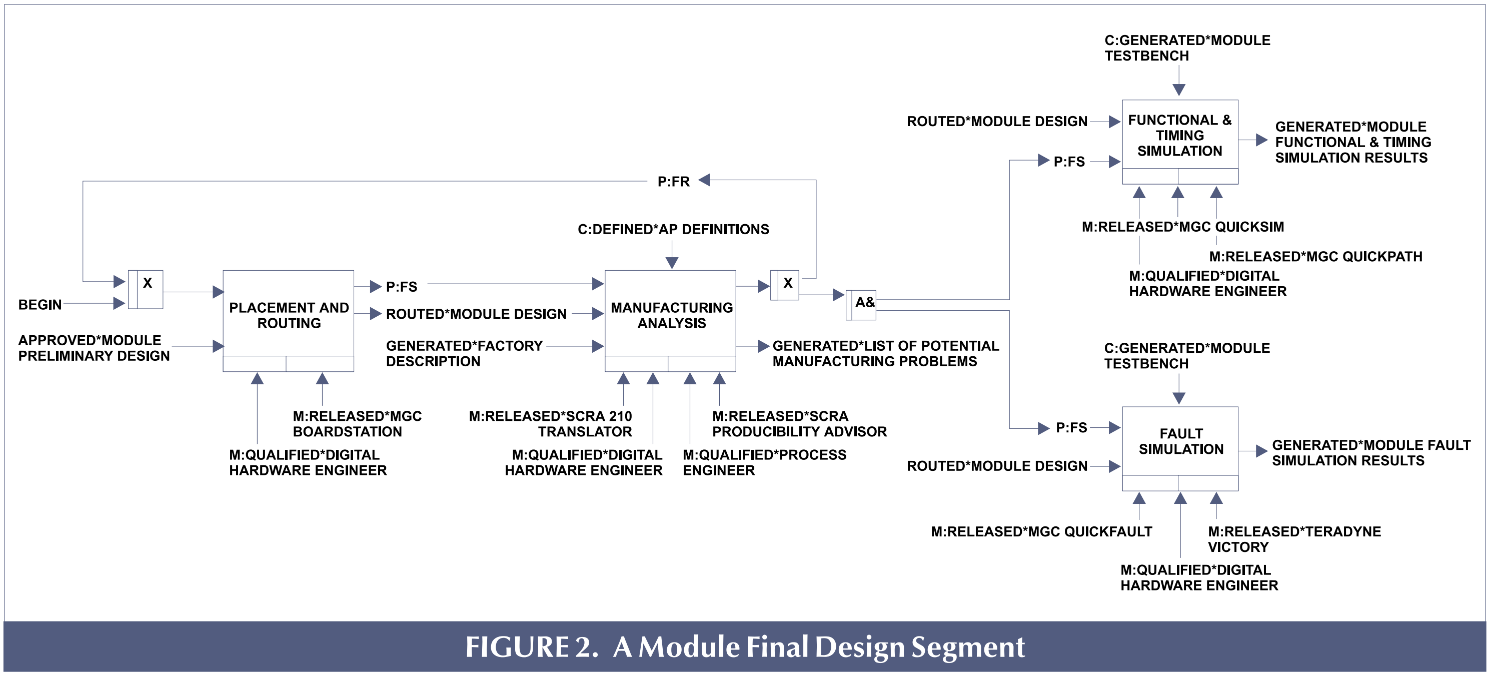

C:GENERATED*MODULE TESTBENCH
ROUTED*MODULE DESIGN
FUNCTIONAL & TIMING SIMULATION
GENERATED*MODULE FUNCTIONAL & TIMING SIMULATION RESULTS
P:FS
P:FR
C:DEFINED*AP DEFINITIONS
M:RELEASED*MGC QUICKSIM
M:RELEASED*MGC QUICKPATH
M:QUALIFIED*DIGITAL HARDWARE ENGINEER
X
BEGIN
APPROVED*MODULE PRELIMINARY DESIGN
PLACEMENT AND ROUTING
P:FS
ROUTED*MODULE DESIGN
GENERATED*FACTORY DESCRIPTION
MANUFACTURING ANALYSIS
X
A&
GENERATED*LIST OF POTENTIAL MANUFACTURING PROBLEMS
C:GENERATED*MODULE TESTBENCH
M:RELEASED*MGC BOARDSTATION
M:QUALIFIED*DIGITAL HARDWARE ENGINEER
M:RELEASED*SCRA 210 TRANSLATOR
M:QUALIFIED*DIGITAL HARDWARE ENGINEER
M:RELEASED*SCRA PRODUCIBILITY ADVISOR
M:QUALIFIED*PROCESS ENGINEER
P:FS
FAULT SIMULATION
GENERATED*MODULE FAULT SIMULATION RESULTS
ROUTED*MODULE DESIGN
M:RELEASED*MGC QUICKFAULT
M:RELEASED*TERADYNE VICTORY
M:QUALIFIED*DIGITAL HARDWARE ENGINEER

FIGURE 2. A Module Final Design Segment

Execution of the workflows is performed using enterprise methodology management tools, which provide links to tools, data access mechanisms, and other services. This process removes these functions as required responsibilities for the design engineer, allowing increased focus on the real design tasks and significantly improving productivity. Using this process, project engineers or supervisors would no longer be responsible for design and implementation of project plans based on workflows using the system.

In addition, the enterprise framework provides multiple workspace views for the design environment to support workflow usage. These views include

- Tool and application workspace
- A data workspace for product and reuse information
- Project/workflow workspaces.

The resources, data objects, and applications available to a particular engineer are defined by his or her identity and role in an authorization hierarchy implemented in the Enterprise System.

2. Workflow Management

Workflow management in the RASSP system is comprised of methods and tools to provide the project team with an environment that facilitates day-to-day work. We have adopted a process-model-driven philosophy for workflow management. The RASSP methodology leverages process models for electronic design that were developed by Lockheed Martin's Engineering Process Improvement program. These models are being augmented with new RASSP process models that specifically address signal processing issues and provide many enhancements to electronics engineering processes. The RASSP methodology also provides approaches for concurrent engineering, evaluation of multiple alternative solutions, failback paths, and iterations.

The detailed representation of the RASSP methodology and related methodologies are modeled using extensions to IDEF3 [2]. The workflow model captures

- Process steps
- Their precedence relationships
- The personnel roles authorized/required to perform work
- The information objects involved (created, used, modified, destroyed, etc.) in the process step
- The tools to be launched or controlled at each step.

The information objects represent place holders for instances of objects that will flow through the workflow. A neutral, process information exchange language (Process Modeling Language - PML) has been developed to facilitate exchange of process data among process-modeling and process-enactment tools. A parser to convert process model data from TopDown Flowcharter to PML format has been developed. The parsed information is stored in a PML repository, which Rockwell is developing. Some implementation details can be found in Lockheed Martin Advanced Technology Laboratories' paper, entitled "Workflow Modeling for Implementing Complex, CAD-Based Design Methodologies" [3].

The workflows are hierarchical in nature — representing the various disciplines associated with electronic design. The workflows consist of reusable workflow segments, which can be combined in various configurations to address specific project needs. Figure 2 represents a module final design segment. These segments consist of multiple process steps, each of which are also reusable. Thus, options

System	Architecture	Detailed Design	Support Workflows
System Requirements Analysis and Refinement	Functional Design	Chassis Design	Reusable Design Element Generation
Functional Analysis	Architecture Selection	Backplane Design	Process Plan Generation
System Partitioning	Architecture Verification	Module Design ASIC Design FPGA Design	Conduct Design Review Reuse Workflow Generation Mentor Part Generation Release to Manufacturing

TABLE 1. RASSP Workflows

available to a user organization are either to make use of the RASSP workflows in current form or to develop process plans based on a combination of reuse of RASSP workflow segments, individual process steps, and possible custom user steps.

The RASSP team is producing workflows that will represent the design disciplines and support activities represented in Table 1. To date, the team has implemented detailed hardware design and multiple architecture design processes. Development of systems and software design processes, as well as enhanced supporting processes, are currently underway.

To support implementation, the detailed workflow information captured in IDEF3X is represented using a workflow-tool-independent language form, the Process Modeling Language (PML). This information can then be transferred to an enterprise workflow tool. Utilization of processes in the enterprise workflow tool involves conversion to an executable form that is compatible with the specific workflow tools being used in the enterprise environment. The Design Methodology Manager (DMM), developed by Intergraph, is the workflow management tool that the Lockheed Martin RASSP system is using for this purpose [4]. In addition to the PML workflow, tool-encapsulation files provide specific tool control information necessary for control of the tool-using workflows in the enterprise environment. This information includes path and name of executables, argument variables, files and data required, pre- and post-processing required, and so on.

3. Information Management

Enterprise information is a key corporate asset and requires a well planned management strategy. The RASSP team developed an enterprise data model, which specifies the metadata that the design engineers and project/system administrators need to track the product and reuse information in the system. In development of the RASSP Enterprise Data Model, several standard models were analyzed relative to RASSP-specific requirements. Models analyzed include the Product Data Control Model (which Rockwell developed on the USAF Integrated Data Strategy program), the STEP parts and protocols AP203 [5], and Part 44 [6]. The Enterprise Data Model was therefore developed based on multiple sources of product data requirements.

The RASSP team is implementing the enterprise data management system using the Intergraph DM2.0 distributed product data management product [7]. As a result, the team is mapping the RASSP enterprise data model to the core model of the DM2.0 product (Figure 3) and is implementing extensions that make practical and commercial sense. Some of the new classes being added to DM2.0 are *security classification*, *anomoly*, *product concept*, and *software configuration item*.

The DM2.0 product manages the enterprise documents and their metadata, product structure and configurations, user roles and authorizations, storage locations and vaults, and related data in a distributed environment. It also interfaces with the reuse libraries to facilitate reuse of the enterprise information. DM2.0 provides these services either directly or under the control of a workflow manager, based on the needs of particular projects. This enables the workflow manager to access and store information (such as design documents, bills of material, and test procedures) by a process step, as needed.

For configuration management [8] and authorization, RASSP-developed models define specific requirements for these capabilities. Support for implementation of these models is provided using the rules subsystem of DM2.0. A combination of DM2.0 and secure internet services will provide distributed product data management capability for a multi-organization, multi-site environment.

4. Reuse Management

Library management in the RASSP system involves the release, cataloging, and searching of reusable design objects. The RASSP Reuse Data Manager (RRDM) supports this library management. Sources for reusable design objects in the RASSP system include:

- CAD tool libraries
- CAD-tool-independent libraries

- Component vendor data books
- Design objects created within a design organization.

In today's design environments, the ability of the design engineer to maximize reuse is impaired because there is no efficient way of searching for reusable design objects across multiple sources, and the various sources of reusable data are not coupled with the design environment. In addition, mechanisms and processes for organizing reusable design objects created within a design organization are lacking. Also lacking is the effective sharing of the reusable design objects within the organization, as well as with other cooperating organizations.

The RASSP Enterprise System will include tools and methods for integrating the various sources of reusable design objects to provide a single source for searching for reusable design data and will enable enterprise-wide sharing of reuse data. The approach consists of

1) Developing a design object class hierarchy, which classifies the various types of design objects in the RASSP domain and models the descriptive data associated with the design objects
2) Developing a commercial library management system, which will implement the design object class hierarchy and provide mechanisms for searching for design objects across multiple libraries and across a virtual enterprise.

A support workflow is also provided by the RASSP system for addition of new reuse elements and/or classes in the system. This process includes certification of the new elements, possibly to the classification hierarchy, and generation of documentation updates.

Additionally, the RASSP reuse management system supports loosely-coupled and tightly-coupled federations of cooperating organizations in sharing library data. The core library management search and browse function, which supports the RASSP design object class hierarchy, was implemented by Aspect Development. This function was released as a commercial product in May 1995. Extensions for integration of reuse library systems are currently in development. An initial version of the reuse class hierarchy is shown in Figure 4. The RRDM extensions being developed support: capabilities to manage default and template objects, manage parametric searches, modify existing objects, modify class hierarchy, and so on.

5. Project Planning

A project plan is a specific collection of workflows that have been customized to meet the specific requirements of a project and the performing organization. The RASSP Enterprise System includes

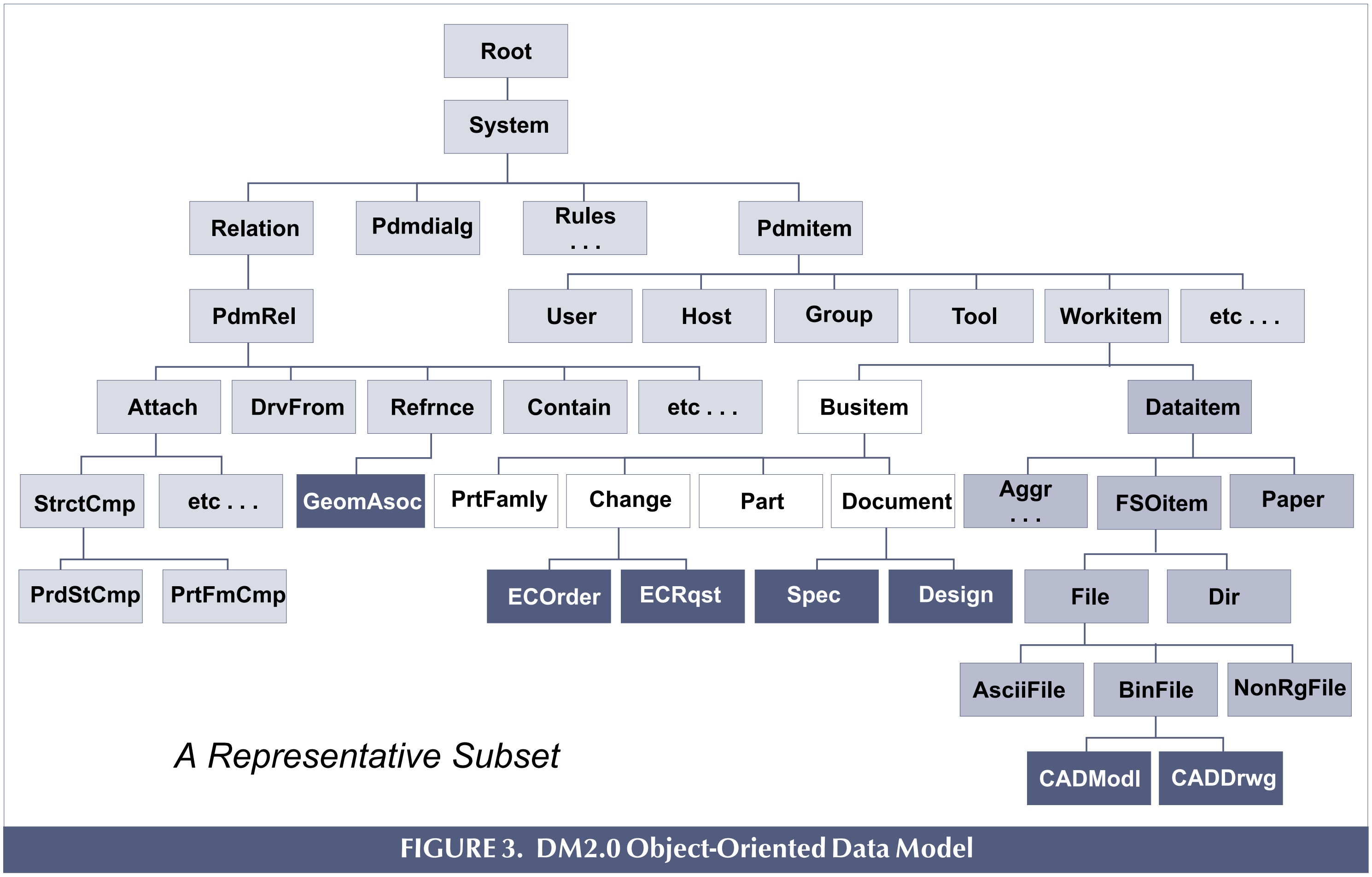


FIGURE 3. DM2.0 Object-Oriented Data Model

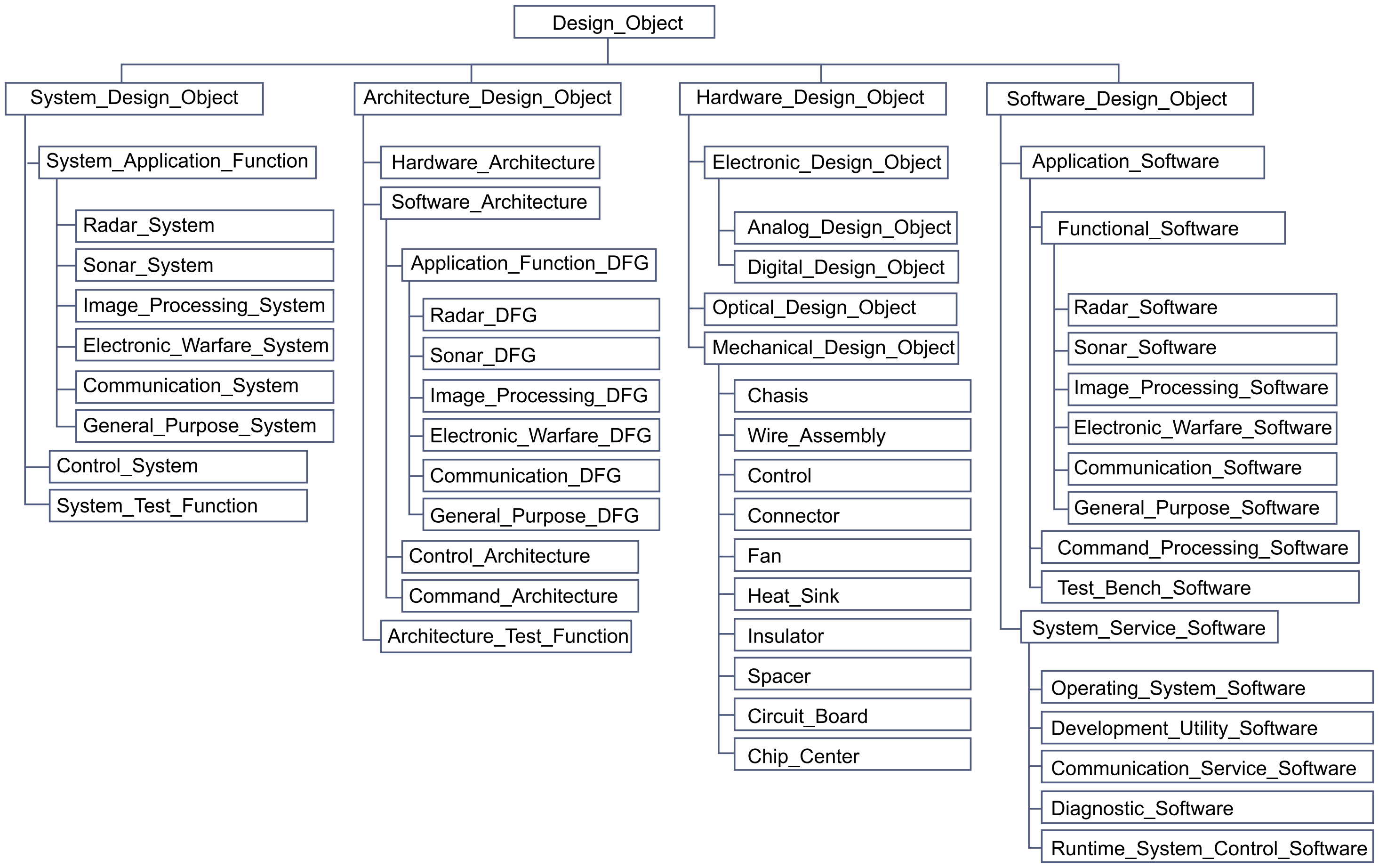


FIGURE 4. RASSP Design Object Class Hierarchy

tools to enable construction of these project plans using workflow segments or other project plans that are maintained in the workflow reuse system (Figure 5). These tools enable selection of the workflow segments, customization of the workflows, linkage of the segments, and definition of totally new workflows. Because the workflows also include the data object definitions, the process of combining workflows into projects produces data object templates. These data object templates specify the detailed information associated with, or produced on, the projects.

Within the RASSP enterprise environment, a project builder toolset that is being developed as an extension of DMM provides capabilities for construction of project plans using the workflow segments or activities, as well as previously developed program plans. These capabilities include the ability to

- Cut and paste workflow models (shallow and deep-copying)
- Browse multiple models
- Interface with reuse repositories
- Capture metadata about models, such as where used, rationale, metrics, author, etc.
- Analyze newly created models for consistency.

The RASSP team anticipates that user organizations will use these tools to create new instances of these workflow models (or even design new workflow models), which are tailored to particular project needs. These models can also be added to the workflow library and made available for use on future projects.

The data object set for a given project represents a set of place holders, or data templates, for management of the project data. These are mapped onto the RASSP information model, which specifies the requirements for management tracking of the data objects. Execution of the workflow steps produces more detailed design information within design objects, such as additional product structure information and/or documentation information.

In execution of the project plan constructed from the workflows, the activities and data object specifications are instantiated for the particular project. As part of this process, users are assigned to the roles in the project plan, and actual object instances are associated with their place holders. The information manager generates the appropriate objects, work locations, and so on to facilitate the workflow. The information models and information manager are therefore closely coupled to the process models and the workflow manager. The harmony between the two enables the users to perform the right tasks using the right information in a transparent fashion.

Summary/Status

Through integration of workflow/process technology and data management of product and reuse information, the RASSP Enterprise System, provides significant capability for enabling large productivity gains for signal processing/electronics engineering teams. The development plan for the RASSP Enterprise System includes four prototype build cycles. The team demonstrated the initial prototype system, which focuses on electronic hardware design, in February 1995. The implementation of the functional design and architecture design processes — which are the focus of the Build 2 system — were demonstrated in February, 1996.

Key benefits include a practical approach to apply workflow technology in an engineering environment, capability for planning and management of complex products involving CAD environment, and improvements in reuse implementation through an integrated, distributed strategy.

Accomplishments to date include definitions of four RASSP workflows; an initial definition of PML, prototypes of a PML parser and PML repository, an initial definition of a reuse hierarchy, extensions to DM2.0 classes, and extensions to RRDM functionality.

References

[1] Martin Marietta, “RASSP First Annual Interim Technical Report,” Moorestown, NJ, 1994.

[2] Armstrong Laboratory, IDEF3 Process Description Capture Method Report, AL-TR-1992-0057, Wright Patterson Air Force Base, OH, 1992.

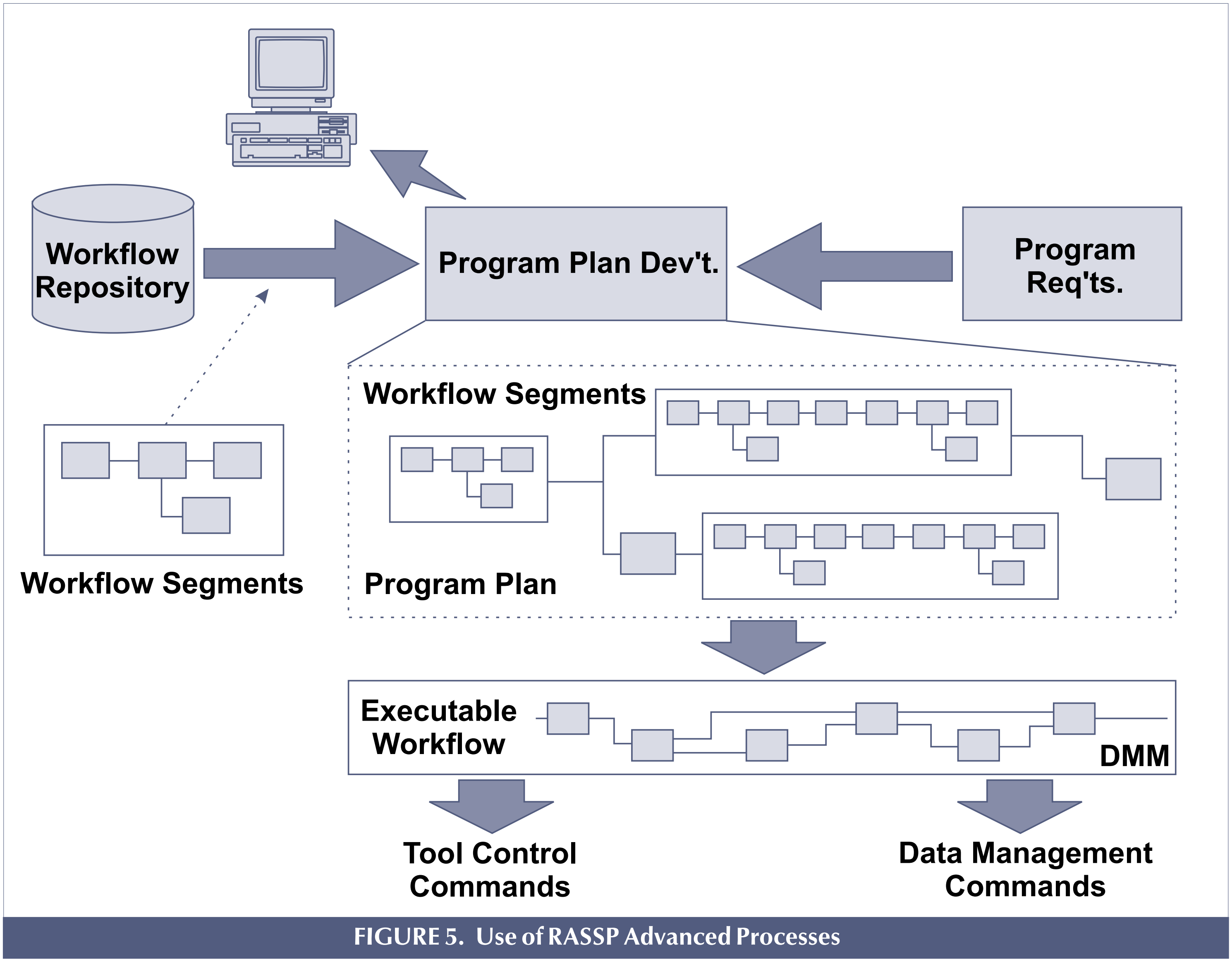


FIGURE 5. Use of RASSP Advanced Processes

[3] Lockheed Martin Advanced Technology Laboratories, "Workflow Modeling for Implementing Complex, CAD-Based Design Methodologies," Camden, NJ, 1995.

[4] Intergraph Corporation, Design Methodology Manager — Users Guide, Huntsville, AL, 1993.

[5] International Standards Organization, Configuration Controlled 3D Designs of Mechanical Parts and Assemblies, ISO 10303-203, Fairfax, VA: U.S. Product Data Association, 1993.

[6] International Standards Organization, Product Structure Configuration, ISO 10303-044, Fairfax, VA: U.S. Product Data Association, 1994.

[7] Intergraph Corporation, DM/Manager — Users Guide, Huntsville, AL, 1995.

[8] Martin Marietta, "The Configuration Management Model for the RASSP System," Moorestown, NJ, 1994.

John Welsh, Biju Kalathil and Bipin Chadha
Lockheed Martin Advanced Technology Laboratories
Camden, NJ
jwelsh@atl.lmco.com, bkalathi@atl.lmco.com,
bchadha@atl.lmco.com

Mary Catherine Tuck and William Selvidge
Intergraph Corporation
Huntsville, AL
mctuck@ingr.com, wselvid@ingr.com

Elisa Finnie
Aspect Development
Mountain View, CA
elisa@aspectdv.com

Arne Bard
Army Research Laboratory
Ft. Monmouth, NJ
abard@ftmon.arl.mil

Enterprise Integration

James Chieks

Abstract

This paper addresses the implementation of the virtual corporation within the context of rapid signal processor design. The paper examines one paper and one presentation, items [1] and [2], written by the Rapid Prototyping of Application Specific Signal Processors (RASSP) Lockheed Sanders Team. The Lockheed Sanders Team comprises four companies, Sanders, Motorola, Hughes and ISX. The Lockheed Sanders team members work as an entity to achieve the RASSP goals of 4X improvement in time-to-market, life-cycle cost and design quality. The references present processes, tools and enabling methodologies allowing team members collaborative, concurrent design of an Infrared Search and Track (IRST) processor. A summary of these references follows. The RASSP Design Environment (RDE) evolution presents advances in enterprise integration. Necessary processes and technologies presented drive further advances toward the virtual corporation.

1. Review

Brief summaries of previous works provide background and a foundation for advancing the virtual corporation. The author encourages the reading of the actual works in conjunction with this paper.

1.1 Collaborative VHDL Modeling Within the RASSP Program Demonstration Project

Reference [1] describes the deployment of a virtual technical design environment employing electronic collocation. The key factor was the choice to use the IEEE 1076 specification for VHDL as an interoperable design language across heterogeneous computing environments. Exchange of VHDL source code supported concurrent development of a design partitioned among three companies. Additional processes and techniques implemented to support multi-site use of VHDL included:

- Operating within an Integrated Product and Process Development Team (IPPDT).
- Daily teleconference meetings to address status.
- Internet based email for data delivery, code debug and coordinative communications.
- Video Conferencing for collaborative program reviews.
- Desktop Conferencing as a collaborative review medium.
- Multi-platform (Sun Microsystems & Hewlett Packard workstations), multi-node, internet-hosted, virtual design database.

- Data Mirroring between design database nodes to preclude bandwidth issues associated with transparent network access.
- Common directory structures among nodes to support data mirroring
- 24 hour availability.
- Background processing in the UNIX operating system for task automation. Use of source code management scripts to maintain data coherency between sites. File Transfer Protocol (ftp) as the bulk data movement utility.
- Tar compression to reduce Internet loading.
- Scripted utilities to automate and standardize the VHDL source code build.
- A Data Promotion scheme to support build dependencies and implement information hiding between sites.
- Design partitioning strategy that minimized complex interaction between sites.
- Common multi-platform design environment software at each team member site.
- Security measures for data encryption.
- Security measures to allow database access only to RASSP team members.

The above strategy resulted in the successful completion of a design and implementation where participants rarely met face-to-face. Reference [1] presents the following obstacles:

- Time zone differences.
- Email delivery delays.
- Visual electronic collocation limitations (slow update rates, limited visibility). The video conferencing environment suffers from bandwidth limitations resulting in less than real time response. As a result of the IRST project, RASSP use of video teleconferencing is limited to
  - Early project contact to establish team rapport.
  - Critical program stages where project direction is being established.
- Internet bandwidth limitations. Due to bandwidth limitations the IRST performed automated bulk data movement of compressed files during non-peak hours.
- Necessity of Non-VHDL graphical methods for communicating design complexities. The IRST project used design tool specific features where it assisted understanding complexities.

Two recommendations from reference [1] include

- Use of ‘make’ files in place of script files for the VHDL build process.
- A VHDL file naming structure that supports configuration control in the build process. Since the IRST project, RASSP developed a VHDL coding style guide that incorporates directory and file naming conventions for collaborative design.

1.2 Achieving Electronic Collocation for Collaborative Work Groups On RASSP

Reference [2] presents several tools needed for electronic collocation (many covered in reference [1]). Specific tools and network architectures are mentioned with an explanation of their function within RASSP. Also introduced is the use of the World Wide Web and email list servers in the RASSP program as alternative data distribution medium. The web contributes greatly to enterprise integration by providing ready access to documents in the Hypertext Markup Language (HTML) as well as providing a graphical interface to documents on the ftp server. The RASSP IPPDT has benefited from the publishing of the RASSP top-down design process. Use of the Web has incorporated enterprise security features such as IP address screening and password-protected access.

Several needs within the RASSP collaborative electronic collocation workspace are addressed within Reference [2] including:

- The need to plan deployment of file server based data and documents.
- Platform independent scalable security.
- The investment and organizational commitment needed or successful electronic collocation.
- The benefit of establishing initial relationships face-to-face prior to electronic collocation.
- The difficulty of implementing project management systems in an electronic collocation workspace.

2. Enterprise Integration in the RDE

RASSP is advancing the enterprise integration with the evolution of the RDE and process development on RASSP. The RDE facilitates Integrated Product and Process Development by providing a collaborative development environment. The RDE supports automating the development process to improve product development, specifically concerning cycle time, product cost, and product quality.

The RDE is part of the RASSP Rolling Wave. The Rolling Wave incorporates the iterative “Model Year” methodology with the IRST distributed development environment representing RDE Model Year 0. Subsequent iterations of the RDE have resulted in a RDE

Prototype (hot mock-up) and RDE Release 1.0. Each iteration incorporates improvements in enterprise integration capabilities.

2.1 RDE Prototype

The RDE Prototype consists of a simulated distributed design tool framework. The prototype serves as a research vehicle. A wide variety of engineers and program managers from three companies experienced the look and feel of a distributed design environment. Among the concepts proposed by the prototype are

- Security: A more robust and secure distributed database.
- Ease of Use: A Graphical User Interface based on the UNIX/ X-Windows standard.
- Distributed Operation: The ability to install client-side utilities and third party tools.
- Electronic Collocation: An integrated collaborative work environment for conducting on-line desktop screen sharing.
- Accessibility: Data file search capabilities.
- Measurability: Automated design metrics capture.

2.2 RDE Release 1.0

The knowledge gained through the development and demonstration of the RDE Prototype proved invaluable in the design and implementation of RDE Release 1.0. The additional enterprise integration concepts incorporated include

- Replication: The RDE operates in beta site installations consisting of government, industry and academic settings. Training provided by RASSP staff acclimates users and administrators. The RDE is delivered on magnetic tape and built on the host processor by RASSP staff.
- Greater Platform Independence was demonstrated.
  - RDE Servers: Sun Sparc w/SunOS, Hewlett Packard.
  - RDA Clients: Sun Sparc, w/SunOS, Macintosh.
- Increased Security: Use of third party application providing DES data encryption (Hughes' 'Netlock').
- Increased Data Hiding: Role assignments and file access privileges.
- Data Tendering: The capture and recording of situational parameters along with data submission.
- Scalability: Supporting large numbers of nodes, clients and tools.

Some issues identified during RDE development reside outside the domain of tool-based solutions. These are people issues. The success of electronic collocation is rooted in the rapport established among the distributed staff. Face-to-face interaction is essential early in the program. Differences in management style and work cultures lower communication bandwidth in an electronic environment. Program management should sponsor an "all-hands" program overview event for the team to get acquainted. Video taping this event acclimates and provides training for employees that join the program later. Initial relationships are re-enforced using video conferencing for early stages of the program. The benefits of video conferencing decrease as working relationships increase the communication bandwidth. Adding structure to daily and weekly operations also increases bandwidth. RASSP teams meet regularly in teleconferences. Structured meeting agendas, meeting minutes and action item tracking improve team awareness and progress.

2.3 Future Requirements

For future tool development, modularity and open architecture design are enabling characteristics. The electronic collocation architecture needs an open interface. This supports third party tool suppliers to develop plug and play modules or to develop data format translators between incompatible tools.

References

[1] Collaborative VHDL Modeling within the RASSP Program Demonstration Project: Ray Dreiling, Paul Kalutkiewicz, Sanders, A Lockheed Martin Company, Mike McCollough Hughes Aerospace & Electronics Company. December 1991.

[2] Achieving Electronic Collocation for Collaborative Work Groups On RASSP, Karen Amestoy Hughes Aircraft, November 8, 1994.

[3] The Extended Enterprise: A descriptive Framework, Some Enabling Technologies and Case studies in the Lotus Notes Environment. Michael Bloch and Yves Pigeur, Ecole des HEC, University of Lausanne, Journal for Strategic Information Systems.

James Chieks
Hughes Aircraft
m/s RE/R01/A508
P.O.Box 92426
Los Angeles CA, 90009-2426
chieks@rassp.hac.com

The National Industrial Information Infrastructure Protocols (NIIIP) Project

Richard Bolton

Abstract

This article describes the NIIIP Consortium, its mission, and technical vision. It provides an overview of the NIIIP reference architecture and its features. It highlights the key state-of-the-art elements of the NIIIP solution, the NIIIP spiral development plan, and the NIIIP deliverables. It summaries the Consortium's accomplishments and future plans. Project information is available at ***http://www.niiip.org****.*

1. Introduction

In today's fast-changing global marketplace, manufacturing organizations need to align themselves closely with both their suppliers and customers. Product cycle times are no longer measured in months and years but in days and weeks. Timeliness and responsiveness are just as important to business success, as are quality, and cost requirements.

Today, the incompatibility of the information technology used by manufacturing organizations, suppliers, customers, and associates, is the major inhibitor to close alignment with new customers and suppliers and to the reduction of cycle times. The goal of the NIIIP project is to solve this incompatibility within Virtual Enterprises and allow organizations to collaborate with each other regardless of data structures, processes, or computing environments.

2. What is the NIIIP?

The NIIIP Consortium consists of a group of thirteen leading United States information technology suppliers and users with a common interest in developing a software architecture and providing technologies to enable Virtual Enterprises. Virtual Enterprises are temporary consortia or alliances of companies formed to exploit fast-changing opportunities. The NIIIP Consortium is national in scope and its members bring a wealth of experience and technology to support Virtual Enterprises. Together with the Federal Government, they share costs and skills to create the necessary infrastructure to support Virtual Enterprises across the United States[1] .

NIIIP Consortium Members include: CAD Framework Initiative, Digital Equipment Corp., Enterprise Integration Technologies, General Dynamics - Electric Boat Division, IBM, International TechneGroup Inc. (ITI), Lockheed Aeronautical Systems, Magavox Electronic Systems, National Institute of Standards and Technology, Rensselaer Polytechnic Institute, STEP Tools Inc., UES Inc., and the University of Florida.

2.1 Mission and Technical Vision

NIIIP's mission is to enable U.S. Industrial Virtual Enterprises to provide globally competitive products, services, and solutions cost-effectively and in a timely manner — regardless of organization, geographic and technical boundaries or company size — and to make U.S. manufacturing the global standard that other nations try to emulate.

NIIIP's technical vision is to define ways for existing applications to inter-operate and to make the technologies fit together in a useful manner based on existing, emerging, and defacto standards. NIIIP focuses on establishing a computer system infrastructure that: makes Virtual Enterprise collaborative computing pervasive among U.S. manufacturers; provides state-of-the-art software technologies to allow participants to effectively collaborate; allows companies within Virtual Enterprises to share costs and skills, and access global markets with each participant contributing its core expertise.

2.2 Reference Architecture

The technology requirements of Industrial Virtual Enterprises include common communication protocols, a uniform object technology base for system and application inter-operability, common information model specifications and exchange, and cooperative management of integrated Virtual Enterprise processes.

The NIIIP reference architecture is distributed, open and non-proprietary software infrastructure that enables Virtual Enterprises to integrate resources and technologies into a production system. As shown in Figure 1, "NIIIP Technology Components," NIIIP is integrating the technologies from four communities to enable the proliferation of Virtual Enterprises across the United States.

NIIIP furthers the adoption and convergence of existing standards and the definition of new ones by working with standards organization such as: the International Standards Organization's Standard for the Exchange of Product Data (ISO-10303 STEP), Internet Engineering Task Force (IETF), the CAD Framework Initiative (CFI), Work Flow Management Coalition (WFMC), Object Management Group (OMG), and others.

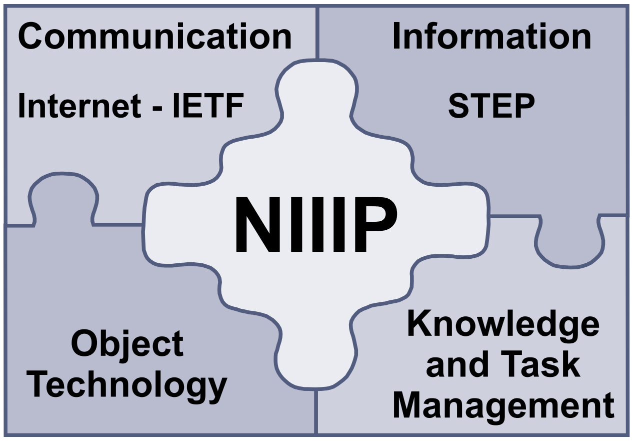


FIGURE 1. NIIIP Technology Components

[1] This research is sponsored by the Advanced Research Projects Agency under ARPA Order No. B761-00 and managed by the United States Air Force under contract F33615-94-2-4447.

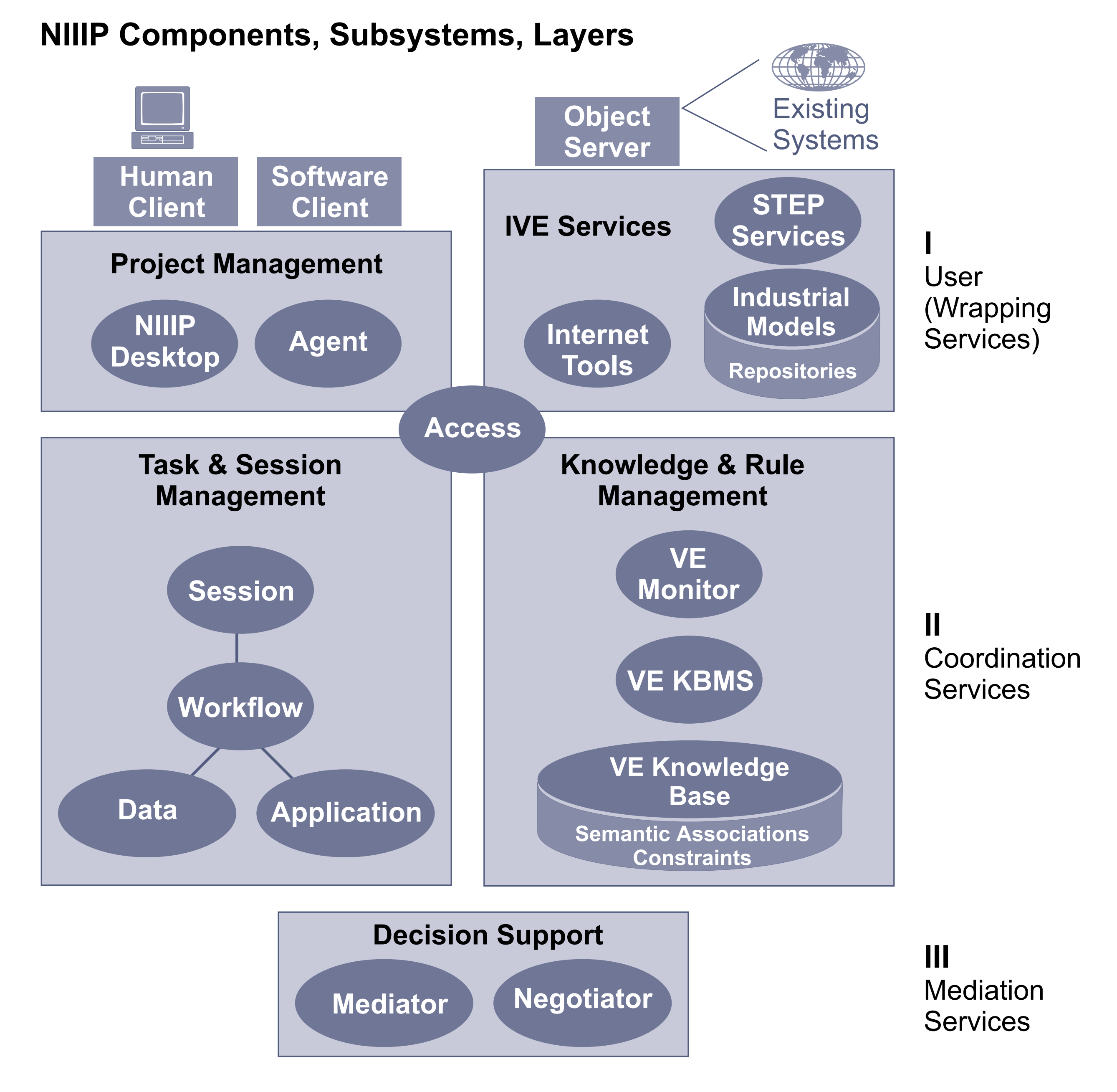


FIGURE 2. The NIIIP Infrastructure

As shown in Figure 2, “The NIIIP Infrastructure,” the NIIIP component protocols consist of a set of thirteen components, their interrelations, and mutual obligations, that allow the formation of various kinds of Virtual Enterprises depending on how many components are selected. These thirteen NIIIP components are packaged into five subsystems positioned across three layers. Currently, a set of system protocols are in the process of being defined that capture common behavior across components and represent a Virtual Enterprise model with both build-time and run-time protocols.

The NIIIP components consist of classes and objects that are described in terms of an amalgam of object modeling paradigms taken mainly from standard sources:

- Interface (CORBA IDL)
- Info Model (ISO-10303 EXPRESS)
- Rules (Event, Condition, Action)
- Constraints (EXPRESS)
- Associations (OMG Relationship Service, OSAM*, etc.)

In Layer I, the user layer or wrapping services layer, end-user applications interface to the NIIIP environment. Layer II, the middle-ware or coordination service layer, provides services to Layer I applications. Layer III, the mediation services layer, provides services to Layers I and II.

The Project Control subsystem consists of Desktop and Agent components. These represent the Virtual Enterprise control function for the end-user or end-user surrogate. The Virtual Enterprise services subsystem contains the design and collaboration tools. They represent the data function for the end-user.

The Task and Session subsystem controls work within the Virtual Enterprise. They might share resources distributed across enterprise boundaries and through various firewalls. The Knowledge and Rules Management subsystem provides the monitoring of Virtual Enterprise rules that allow inter-enterprise resource sharing. The Layer III Decision Support subsystem components help to resolve faulty requests, provide for the acquisition of new knowledge, and provide for negotiation between agents in dispute.

2.3 Components

The thirteen NIIIP components are summarized in Table 1. Internal to each component are one or more objects (OMG "interfaces"). No single NIIIP-compliant product will likely support all the behaviors specified by the thirteen components. Many products will, however, function as instances of NIIIP-compliant classes generated during the NIIIP conformance testing process.

NIIIP will make its architecture public, and will deploy its technologies nationwide so that organizations can adopt the technology and apply it to their particular situations.

2.4 State-of-the-art Technology

The NIIIP protocols provide an infrastructure for the inter-operation of commercial-off-the-shelf (COTS) products in the industrial domain.

In the NIIIP environment, resources of member organizations such as database systems, product data files, expert systems, application systems, are uniformly modeled as components in an object-oriented framework. Key elements of the NIIIP solution include

- the NIIIP Common Language (NCL) that allows Virtual Enterprise implementors to model resources and associated protocols. This language is an optional superset of DMG IDL and ISO EXPRESS and can be used directly,
- a distributed Virtual Enterprise monitor for ensuring that business and security rules are enforced,
- middleware for secure Internet access and inter-operability across firewalls,
- NIIIP tools for uniform real and simulated work management solutions to allow Virtual Enterprise members to pre-plan and validate the flow of information and sequence appropriate technical and business review processes that support the manufacturing cycle, and
- protocols for mediated and negotiated data and process interchanges based on the Virtual Enterprise knowledge base.

Meta-class	Protocol Areas
NIIIP Desktop	User interface to multiple VE, sessions, roles, VE administration
Access	Both CORBA and Non-CORBA mediated object access; replication, caching, DFS, SHTTP, TCP/IP
Agent	Creates VE organizational structures, resource ownership, settlement, transducers
STEP Services	Behaviors to support the STEP Data Access Interface (SDAI) for VE Industrial Models
Data Mgmt	Shared data management: query, change management; configuration management; view/schema/encryption translations between source and target
Workflow	Cross-product workflow management
Session	Authentication, resource allocation, transaction logging; simulation; long transaction control
Application/Tool	Private data management; application invocation setup; moving data to application to data; tool-talking
Mediator	Resolution of semantic ambiguity; summarizing, abstracting, validating data; localized knowledge; maintenance of local ontologies
Negotiator	Impartial management of multi-party negotiations
VE KBMS	Active semantic network database for VE
Internet Tools	Access to the services of the Internet; conferencing; WWW; control point for the secure VE, firewall services
VE Monitor	Trap CORBA requests; inspects "context"; transparently dispatch NIIIP infrastructure components; enforces business rules

TABLE 1. NIIIP Components

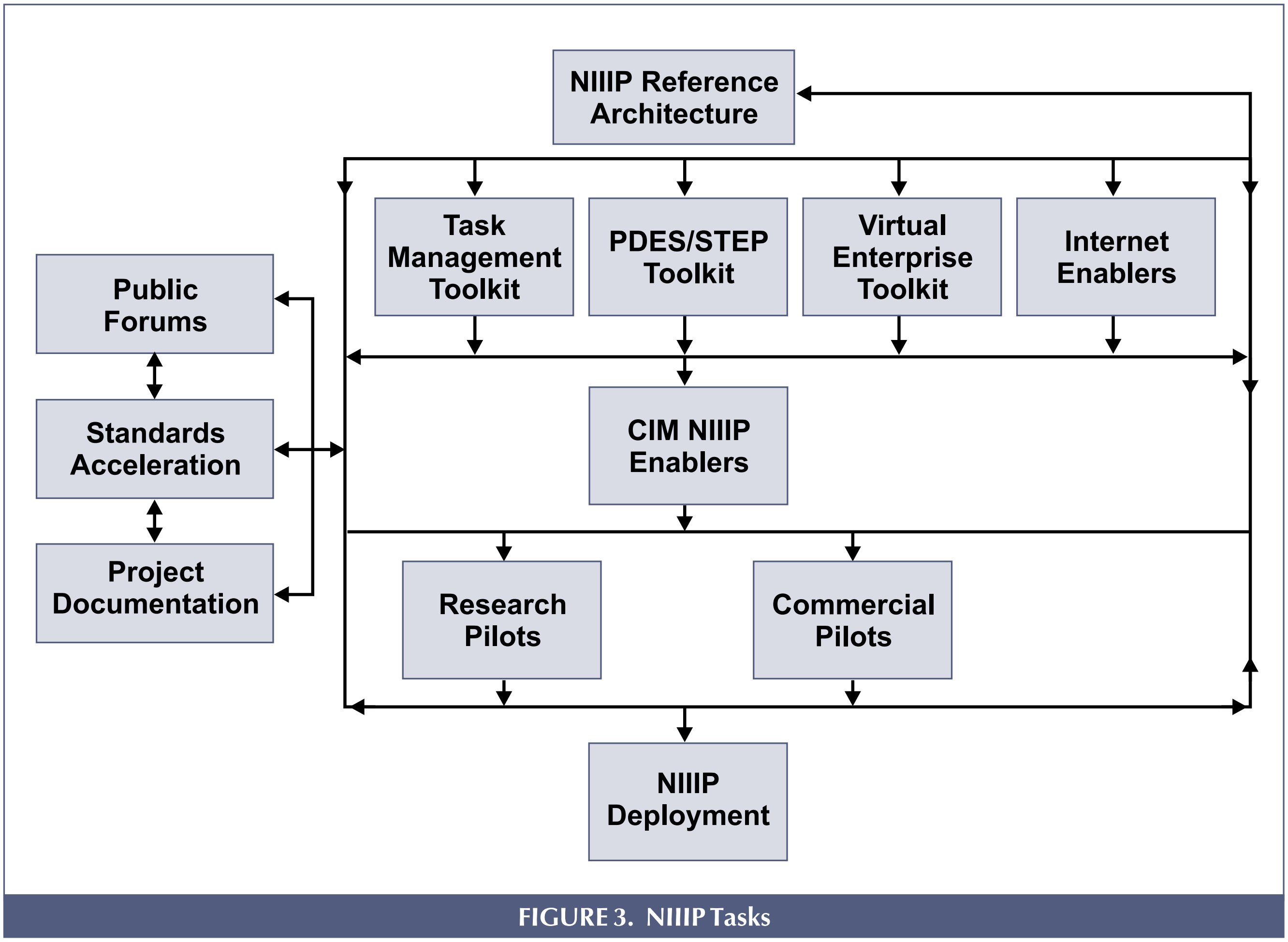


FIGURE 3. NIIIP Tasks

2.5 Spiral Development

To minimize technical risks, NIIIP's spiral development plan includes iterative development, incremental increases in function, and use of rapid prototyping tools.

The NIIIP consortium, as show in Figure 3, "NIIIP Tasks," has twelve tasks. Its Reference Architecture drives the development of Protocols that allow experimentation with various subsets of the technology across the Internet. Protocols allow the construction of commercial, defense, and research pilots. The Consortium will accelerate standards, deployment, and commercialization.

3. NIIIP Accomplishments and Plans

Cycle 1

During this cycle, NIIIP instantiated its protocols with emphasis on Task, Session, Workflow and Data (including STEP) Management with CORBA over the Internet. The Consortium demonstrated the initial infrastructure and documented the experiences learned. These experiences are incorporated into subsequent NIIIP Cycles.

Cycle 2

During this cycle, NIIIP will partially implement its protocols — component and system — with emphasis on the Knowledge Base, Monitor, Mediation, and Agents with CORBA over the Internet. Workflow and ORB interoperability will also be included.

Cycle 3

During this cycle, NIIIP intends to complete an implementation of the protocols defined during cycles 1 and 2. This includes the use of both system and component protocols.

Acknowledgments

The author wishes to recognize the following individuals for their help in the preparation of this article: Esther Odescalchi, Art Goldschmidt, Alex Putman, and David Zenie.

Richard Bolton
National Industrial Information
Infrastructure Protocols (NIIIP) Consortium
1055 Washington Boulevard
Stamford, CT   06901
director@niiip.org

Concurrent Engineering Wheels

Biren Prasad

Abstract

The Concurrent Engineering approach to product design and development has two major themes. The first theme is establishing a concurrent product and process organization*. This is referred to herein as* "process taxonomy." *The second theme is applying this process taxonomy (or methodology) to design and develop the total product system. This is referred to as* integrated product development (IPD)*. Each theme is divided into several essential parts forming major arms of the so called concurrent engineering wheel-set [Prasad, 1996a]. This article describes a Concurrent Engineering (CE) wheel-set and explains the basic principles on which this very subject is founded.*

The first theme called product and process organization (PPO) *has nine arms. The second theme named integrated product development (IPD) has ten arms. The materials in these two CE themes are brought together to balance the interests of both the customers and the companies. The arms of the PPO theme are* Life-cycle Management: Process Re-engineering, Cooperative Work-groups, System Engineering, Information Modeling, The Whole System, *and* Product Realization Taxonomy. *The arms of the IPD theme are* Total Value Management, CE Metrics and Measures, Concurrent Function Deployment, Product Development Methodology, Decision Support Systems, Intelligent Information System, Capturing Life-Cycle Values, Life-Cycle Mechanization, *and* IPD Deployment Methodology *[Prasad, 1996b].*

In the Concurrent Engineering (CE) system, each modification of the product realization represents a taxonomic relationship between specifications (inputs, requirements and constraints), outputs, and the concept it represents [ASME/NSF, 1996]. At the beginning of the design process, the specifications are generally in abstract forms. As more and more of the specifications are satisfied, the product begins to take shape (begins to transform into a physical form). To illustrate how a full CE system will work, and to show the inner-working of its elements, the author defines this CE system as a set of two synchronized wheels. The representation is analogous to a set of synchronized wheels on a bicycle. Figure 1 shows this CE wheel set.

1. CE Wheel Set

The first CE wheel represents the *integrated product and process organization*. The second CE wheel accomplishes the *integrated product development*. The two wheels together harmonize the interests of the customer and the fostering CE organization (frequently referred to as an enterprise). Three concentric rings represent the three essential elements of a wheel. The middle ring represents the CE work-groups, which drive the customer and the enterprise the way a human drives a bike. The work-groups are divided into four quadrants representing the four so-called CE teams. These teams are: *the personnel team, the technology team, the logical team* and the *virtual team*. The outer ring for each wheel is divided into eight parts. The arms for the first wheel constitute the PPO theme. The PPO theme explains how a CE design process (referred to herein as CE process taxonomy) provides a stable, repeatable process through which increased accuracy is achieved. The PPO theme starts with manufacturing competitiveness reviewing history and emerging trends. The remaining parts of the PPO theme describe the CE design process, explain how concurrent design process can create a competitive advantage, describe CE process taxonomy, and address a number of major issues related to *product and process organization.* The arms of the second wheel constitute the IPD theme.

2. First CE Wheel: Integrated Product and Process Organization

The innermost ring of the first CE wheel is a hub. The layout of the hub is the same for both wheels. The hub represents four supporting "M" elements: *models, methods, metrics and measures*. Models refer to *information modeling.* Methods refer to *product realization taxonomy*. They are part of the PPO theme. *CE Metrics and Measures* are part of the IPD theme. The complexity of the product realization process (PRP) [NSF/ASME, 1996] differs depending upon the (i) types of information and sources, (ii) complexity of tasks, and (iii) the degree of their incompleteness or ambiguity. Other dimensions encountered during this PRP that cannot be easily accommodated using traditional processes (such as serial engineering) are: (iv) timing of decision making, (v) order of decision making, and (vi) communication mechanism. The elements of the first CE wheel define a set of systems and processes that have the ability to handle all of the above six dimensions. In the following section some salient points of the arms are briefly highlighted.

- *Manufacturing Competitiveness*: The price of the product is dictated by world economy and not by a country's economy or a company's market edge alone. Those companies that are global can quickly change to suit a changing world market place and position themselves to compete globally rather than locally. This arm outlines what is required to become a market leader and compete globally. Successful companies have been the ones who have gained a better focus on eliminating waste (which normally would slip into their products), by understanding what drives product and process costs and, how value can be added. They have focused on product and process delivery-system — how to transition process innovations into technical success and how to leverage the implementation know-how into big commercial success. Many have chosen to emphasize high-quality flexible or agile production in product delivery rather than high-theme (mass) production.

- *Life-cycle Management*: Today, most companies are under extreme pressure to develop products within time periods that are rapidly shrinking. As the market changes, so do the

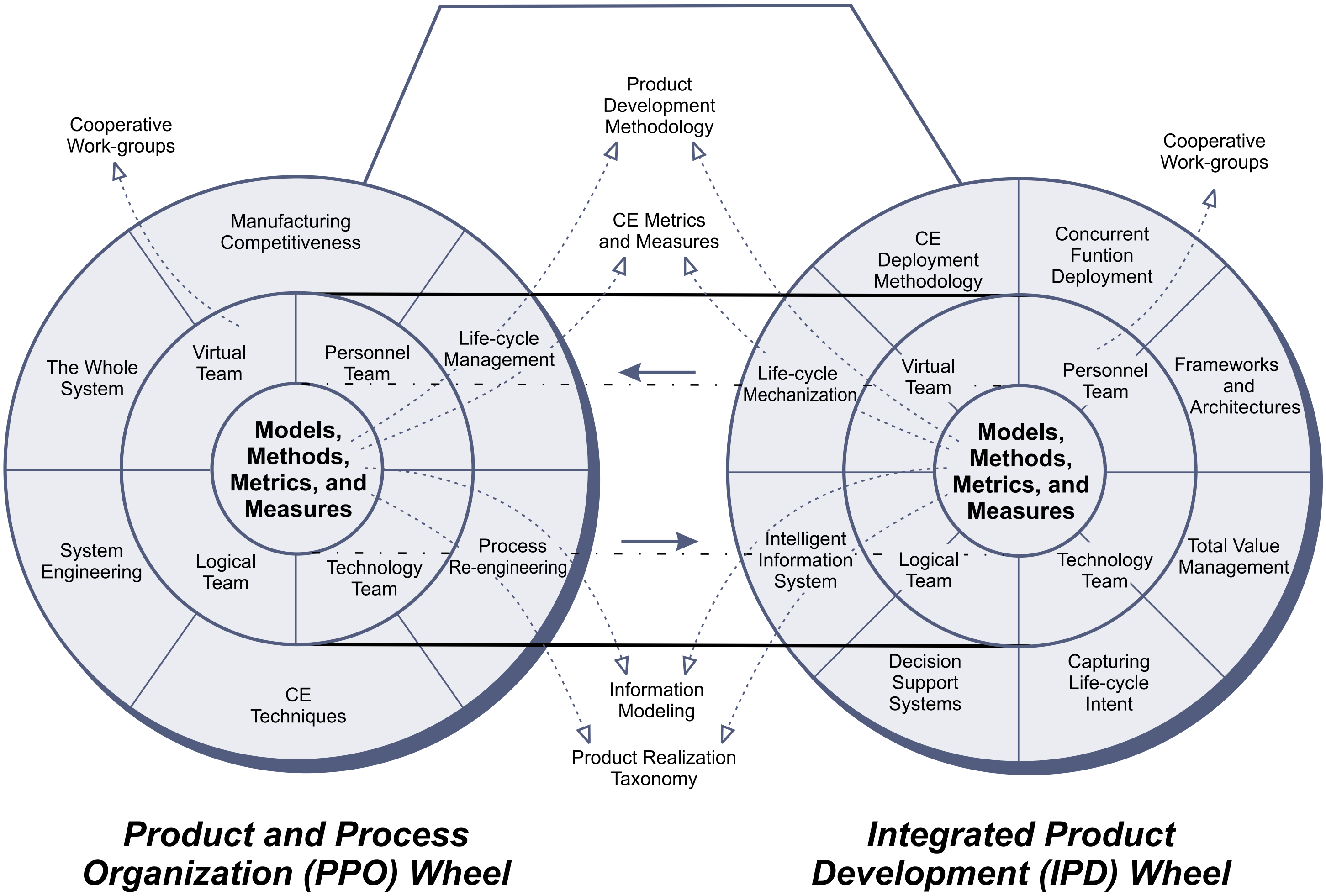

Product Development Methodology
Cooperative Work-groups
Cooperative Work-groups
CE Metrics and Measures
Manufacturing Competitiveness
The Whole System
Virtual Team
Personnel Team
Life-cycle Management
Models, Methods, Metrics, and Measures
System Engineering
Logical Team
Technology Team
Process Re-engineering
CE Techniques
Information Modeling
Product Realization Taxonomy
CE Deployment Methodology
Concurrent Funtion Deployment
Life-cycle Mechanization
Virtual Team
Personnel Team
Frameworks and Architectures
Models, Methods, Metrics, and Measures
Intelligent Information System
Logical Team
Technology Team
Total Value Management
Decision Support Systems
Capturing Life-cycle Intent
Product and Process Organization (PPO) Wheel
Integrated Product Development (IPD) Wheel

FIGURE 1. A Synchronized Set of CE Wheels

requirements. This complicates the management of continuously varying product specifications and the handling of ongoing changes within this shrinking time period. The ongoing success of an organization lies in its ability to: continue to evolve; quickly react to changing requirements; reinvent itself on a regular basis; and keep up with ever-changing technology and innovation. Many companies are stepping up the pace of new product introduction, and are constantly learning and embracing new ways of engineering products more accurately the first time, and more often thereafter. This arm outlines life-management techniques, such as change management and process improvement, to remain globally competitive.

- *Process Re-engineering*: The global marketplace of the 1990s has shown no sympathy to tradition. The reality is that if the products manufactured do not meet the market needs, demand declines and profits dwindle. Many companies are finding that true increases in productivity and efficiency begin with such factors as clean and efficient processes, good communication infrastructure, teamwork, and a constancy of shared vision and purpose. The challenge is simply not to crank up the speed of the machine so that its outputs (per unit of time) are increased, but to change the basic machinery or process that produces the outputs. To accomplish the latter goals, this arm describes several techniques to achieve competitive superiority such as benchmarking, CPI, organizational restructuring, renovation, process re-engineering, etc.

- *CE Techniques*: The changing market conditions and international competitiveness are making the time-to-market a fast shrinking target. Over the same period, the diversity and complexity of the products have increased multi-fold. Concurrency is the major force of Concurrent Engineering.

Paralleling describes a "time overlap" of one or more work-groups, activities, tasks, etc. This arm describes seven CE principles to aim at: *Parallel work-group; Parallel Product Decomposition; Concurrent Resource Scheduling; Concurrent Processing; Minimize Interfaces; Transparent Communication; and Quick Processing*; This arm also describes the seven forces that influence the domain of CE as agents (referred to here as 7Ts) namely: *talents, tasks, teams, techniques, technology, time and tools.*

- *Cooperative Work-groups*: It has been a challenge for the design and manufacturing engineers to work together as teams to improve quality while reducing costs and lead-time. A single person, or a team of people, is not enough to provide all the links between human knowledge and skills, logical organization, technology, and the set of 7Cs coordination attributes. A number of supporting teams are required, some either virtual or at least virtually collocated. For the waltz of CE synthesis to succeed, CE teams need clear choreography. This arm describes for the first time the four collaborative teams that are essential for managing a CE organization. Examples of collaborative features include capabilities of electronic meeting, such as message-posting and interactions through voice, text, graphics and pictures.

- *System Engineering*: Most groups diligently work to optimize their subsystems, but due to a lack of incentives they tend to work independently of each other. This results in a product which is often sub-optimized at each decomposed level. System engineering requires that product realization is viewed as a "system-centered" problem, as opposed to "component-centered." *Systems Engineering* does not disagree with the idea of compartments or divisions of works, but it emphasizes that the interface requirements between the divisions (inter-divisional) and across the level should be adequately covered. That way, when the time comes to modernize other components of the system, one has the assurance that previously introduced technologies and processes will work logically in a fully-integrated fashion, thereby increasing net efficiency and profitability.

- *Information Modeling*: A successful integrated product development (IPD) requires a sufficient understanding of the product and process behaviors. One way to achieve this understanding is to use a series of reliable information models for planning, designing, optimizing and controlling each unit of the IPD process. The demands go beyond 3-D CAD geometric modeling. The demands require schemes that can model all phases of a product's life-cycle from cradle to grave. The different aspects of product design (planning, feasibility, design, process-planning), process design (process-execution, production, manufacturing, product support), the human behavior in teamwork, and the organization or environment in which it will operate, all have to be taken into account. Five major classes of modeling schemata are defined:

  (a) Product representation schemes and tools for capturing and describing the product development process and design of various interfaces, such as design-manufacturing interface;

  (b) Schemes for modeling physical processes, including simulation, as well as models useful for product assessments, such as DFA/DFX, manufacturability evaluation of in-progress designs;

  (c) Schemes for capturing (product, process, and organization structure) requirements or characteristics for setting strategic and business goals;

  (d) Schemes to model enterprise activities (data and work flow) in order to determine what types of functions best fit the desired profitability, responsiveness, quality and productivity goals; and

  (e) Schemes to model team behavior, because most effective manufacturing environments involve a carefully orchestrated interplay between teams and machines.

- *The Whole System*: While designing an artifact, work-groups often forget that the product is a system. It consists of a number of subassemblies, each fulfilling a different and distinct function. A product is far more than the collection of components. Without a structure or "constancy-of-purpose" there is no system. The central difference between a CE transformation system and any other manufacturing system, such as serial engineering, is the manner in which the tasks' distribution is stated and accomplished. In a CE transformation system, the purpose of every process step of a manufacturing system is not just to achieve a transformation, but to accomplish this in an optimal way. This arm proposes a system-based taxonomy, which is founded on parallel scheduling of tasks and breakdown structures for product and process, and work to realize a drastic reduction in time and cost in product and process realizations.

- *Product Realization Taxonomy*: This constitutes a "state of series of transformation" leading to a complete, or mature design. *Product Realization Taxonomy* involves items related to design completeness, product development practices, readiness feasibility, and quality assessment. In addition, CE requires these taxonomies to have a unified "product realization base." The enterprise integration metrics of the CE model should be well characterized, and the modeling methodologies and/or associated ontology for developing them should be adequate for describing and integrating enterprise functions. The methodologies should have built-in product and service accelerators. Taxonomy is comprised of the product, process descriptions, classification techniques, information concepts, representation, and transformation tasks (inputs, requirements, constraints and outputs). Specifications describing the transformation model for product realization are included as part of the taxonomy descriptions.

3. Second CE Wheel: Integrated Product Development

The second CE wheel defines integrated product development (IPD). IPD in this context does not imply a step-by-step serial process. Indeed, the beauty of this wheel (integrated product development) is that it offers a framework for a concurrent product design and development. A Framework within which the CE teams have flexibility to move about, fitting together pieces of the puzzle as they come together. CE teams have an opportunity to apply a variety of techniques contained in this theme (such as:*Concurrent Function Deployment*, *Total Value Management*, *Metrics and Measures*, etc.), and to achieve steady overall progress towards a finished product.

- *Concurrent Function Deployment*: The role of the organization and the engineers is changing today, as is the method of doing business. Competition has driven organizations to consider concepts such as time compression (fast-to-market), Concurrent Engineering, Design for X-ability, and Tools and Technology (such as Taguchi and Value Engineering), while designing and developing an artifact. Quality Function Deployment (QFD) addresses major aspects of "quality" with reference to the functions it performs, but this is one of the many functions that need to be deployed. With conventional deployment, it is difficult, however, to address all aspects of Total Values Management (TVM) such as *X-ability*, *Cost*, *Tools and Technology*, *Responsiveness and Organization* issues. It is not enough to deploy just the "*Quality*" into the product and expect the outcome to be*World Class*. TVM efforts are vital in maintaining a competitive edge in today's world marketplace.

- *CE Merits and Measures*: Metrics are the basis of monitoring and measuring process improvement methodology and managing their effectiveness. Metric information assists in monitoring team progress, measuring quality of products produced, managing the effectiveness of the improved process, and providing related feedback. Individual assurances of DFX specifications (one at a time) do not capture the most important aspect of Concurrent Engineering — the system perspectives, or the trade-offs across the different DFX principles. While satisfying these DFX principles in this isolated manner, only those which are not in conflict are usually met. Concurrent engineering views the design and evaluates the artifact as a system, which has a wider impact than just sub-optimizing the sub-systems within each domain.

- *Total Value Management*: The most acclaimed slogan for introducing a quality program in early corporate days was simply to provide the *most value for the lowest cost*. This changed as competitiveness became more fierce. For example, during the introduction of the traditional TQM program in 1990 "*getting a quality product to market for a fair price*" was the name of the game. The new paradigm for CE now is TVM: "*to provide the total value for the lowest cost in the least amount of time, and provide what satisfies the customers the most while generating a fair profit for the company*." Here use of value is not just limited to "*quality*." To provide long lasting added value, companies must change their philosophy towards things like *X-ability*, *responsiveness*, *functionality*, *tools and technology*, *cost*, *architecture*, etc.

- *Product Development Methodology*: A systematic methodology is essential in order to be able to integrate: (a) teamwork; (b) information modeling; (c) product realization taxonomy; and (d) measures of merits (called CE metrics), and quantitatively assess the effectiveness of the transformation. This may involve identification of performance metrics for measuring the product and process behaviors. Integrated product development methodology is geared to take advantage of the product realization taxonomy.

- *Frameworks & Architectures*: In order to adequately support the CE 4Ms (namely: modeling, methods, metrics and measurements), it is necessary to have an architecture that is openly accessible across different CE teams, information systems, platforms, and networks. Architecture consists of information contents, integrated data structures, data states, behavior and rules. An architecture not only provides an information base for easy storage, retrieval, and version control tracking, it can also be accessed by different users simultaneously, under ramp-up scheduling of parallel tasks, and in synchronization. We also need a product management system containing work management capabilities integrated with the database. This is essential because in CE there exists a large degree of flexibility for parallelism that must be managed in conjunction with other routine file and data management tasks.

- *Capturing Life-cycle Intent*: Most C4 tools are not really "capture" tools. In static representation of CAD geometry, configuration changes cannot be handled easily, particularly when parts and dimensions are linked. This has resulted in loss of configuration control, proliferation of changes to fix the errors caused by other changes, and sometimes ambiguous designs. By capturing "design intent" as opposed to "static geometry," configuration changes could be made and controlled more effectively using the power of the computer than through traditional CAD attributes (such as lines and surfaces). The power of a "capture" tool comes from the methods used in capturing the design intent initially so that the required changes can be made easily and quickly if needed. "Life-cycle capture" refers to the definition of the physical object and its environment in some generic form. "Life-cycle intent" means representing the life-cycle capture in a form, which can be modified and iterated until all the life-cycle specifications for the product are fully satisfied.

- *Decision Support System*: In CE, cooperation is required between CE teams, management, suppliers, and customers. A knowledge-based support system will help the participating teams in decision making and reflecting balanced views. Tradeoffs between conflicting requirements can be made on the basis of information obtained from sensitivity, multi-criterion objectives, simulation, or feedback. The taxonomy can be made a part of the decision support system (DSS) in supporting decisions about product decomposition by keeping track of what specifications are satisfied, ensuring common visibility in the state of product realization, including dispatching and monitoring of tasks, structure, corporate design histories, etc.

- *Intelligent Information System* (IIS): Another major goal of CE is to handle information intelligently in multi-media—audio, video, text, graphics. Since IIS equals CIM plus CE, with IIS, many relevant CE demands can be addressed and quickly processed. Examples include (a) over local or wide area networks, such as SQL, which connects remote, multiple databases and multimedia repositories; (b) any needed information, such as recorded product designers' design notes, figures, decisions, etc., can be made available on demand at the right place at the right time; (c) any team can retrieve information in the right format and distribute it promptly to the other members of the CE teams.

- *Life-cycle Mechanization*: Life-cycle mechanization equals CIM + Automation + CE. Life-cycle mechanization is arranged under a familiar acronym: CAE, for CIM, Automation, and CE. Since CAE also equals IIS plus automation, the major benefits of mechanization in CAE come from removing or breaking barriers. The three common barriers are: (a) integration (this is a term taken from CIM), (b) automation, and (c) cooperation (which is a term taken from CE). CE provides the decision support element, and CIM provides the framework & architecture plus the information management elements. *Life-cycle Mechanization* refers to the automation of life-cycle functions or creation of computerized modules that are built from one another and share the information from one another. This includes integration and seamless transfer of data between commercial computer-based engineering tools and product-specific in-house applications. This tends to reduce the dependency of many CE teams on communication links and product realization strategies, such as decomposition and concatenation.

- *CE Deployment Methodology*: The purpose of this arm is to offer an implementation guideline for product redesign and development through its life-cycle functions. IPD implementation is a multi-track methodology. The tracks overlap, but still provide a structured approach to organizing product ideas and measures for concurrently performing the associated tasks. Concurrency is built in a number of ways, depending upon the complexity of the process or the system involved. This arm proposed a set of *"Ten Commandments,"* which serves to guide the product and process iterative aspects of IPD rather than just the work-group collaborative aspects during the development cycle. The CE teamwork in the center of the wheel ensures that both local or zonal iterative refinements and collaborative refinements take place during each concurrent track.

4. A Synchronized Wheel Set for CE

All the above arms of CE put together create a synchronized wheel set for CE, as shown in Figure 1. The teamwork, with four cooperating components (technological teams, logical teams, virtual teams, and personnel teams), is in the inner circle. The 4Ms (models, metrics, measurements and methodology) form the center of this wheel. It has four arms to it: *Information Modeling; Product Realization Taxonomy; Measures of Merit; and Integrated Product Development*. The 4Ms are shown in the center because they provide the methodology for guiding the product realization process. The two inner rings, which are the same for both wheels, make the wheels a synchronized set. The teams in the inner circle are the driving force of the methodology (listed in the center) and controller of the technologies (listed on the outer circle). *The emphasis of a team-centered wheel for CE is a departure from a conventional function-centered approach*. Outer circles of each wheel contain the remaining arms of *integrated product and process organization* (PPO theme) and *integrated product development* (IPD theme), respectively. The idea of this inner circle is to provide team-centered 7Cs (*Collaboration*, *Commitment*, *Communications*, *Compromise*, *Consensus*, *Continuous Improvement*, and *Coordination*) interplay across layers of enabling technologies and methodologies. Everything is geared towards cutting and compressing the time needed to design, analyze, and manufacture marketable products. Along the way, costs are also reduced, product quality is improved and customer satisfaction is enhanced due to the synchronized process. There is, however, a finite window in which the benefits of time compression and cost cutting are available. As more manufacturers reduce lead time, what once represented a competitive advantage can become a weakening source. Fortunately, the CE wheel-set provides a continuum (dynamic) base through which new paradigms (process, tools and technology) can be launched to remain globally competitive for the long haul.

5. Major Attributes of this Synchronized Wheel-Set

Whether you are a firm CE believer or not, this dual wheel set provides a complete view of CE from all aspects and perspectives. The management perspective, which is a part of the philosophical aspect, relates to organization and culture. The wheel-set articulates major CE aspects by illustrating the differences between the best methodologies (and taxonomies) from what is currently being practiced.

Examples of major attributes incorporated in the dual wheel-set are:

- Eight fundamental principles on which CE is founded
- Seven primary components of concurrency and simultaneity
- CE environment and its five essential components
- Seven C's to ensure cooperation among work-groups
- Seven primary influencing agents (called 7Ts) for achieving concurrency and simultaneity.
- Cooperative work-group environment spanned by four concurrent teams: (namely — logical team, personnel team, virtual team and technological team)

The first wheel (PPO theme) deals with process taxonomy for CE. Process taxonomy is necessary to adequately classify, distribute and distinguish differences in behaviors of complex enterprise integration systems. The innermost core of this process taxonomy is its foundation, which has four supporting "M elements": models, methods, metrics and measures as mentioned earlier.

References

[1] Prasad, B., 1996a, *Concurrent Engineering Fundamentals*, Volume I: Integrated Product and Process Organization, New Jersey: PTR Prentice Hall.

[2] Prasad, B., 1996b, *Concurrent Engineering Fundamentals*, Volume II: Integrated Product Development, New Jersey: Prentice Hall, (in Press).

[3] ASME/NSF, 1995, *Mechanical Engineering Curriculum Development Initiative: Integrating the Product Realization Process (PRP) into the Undergraduate Curriculum*, New York: American Society of Mechanical Engineering.

Dr. Biren Prasad
Automated Concurrent Engineering
Electronic Data Systems
DELPHI Automotive Systems
1401 Crooks Road, Troy, MI 48084
bprasad@cmsa.gmr.com

Agility through Information Sharing: Results Achieved in a Production Environment

Anthony J. Gadient, Lynwood E. Hines, John Welsh, Andrew P. Schwalb

Abstract

As organizations seek to improve their competitive position by responding effectively to the increasing rate of change in the market place, the need for agile enterprises and agile manufacturing has come to the forefront. In this paper we examine the essential role that information sharing plays in enabling the agile manufacturing of complex products. The principle goal of an agile manufacturing interface is to provide the information sharing infrastructure necessary to enable the formation of virtual organizations and to provide them with the robust DFx (Design for x, where x = producibility, testability, maintainability, etc.) mechanisms they need in order to develop high quality products in a timely, cost effective manner. Results achieved from the implementation of an agile manufacturing interface in a production environment are highlighted. These results include a 10x reduction in the cycle-time required to go from design to manufacturing set-up, and a reduction in the rework for complex Printed Circuit Assemblies of up to 80%.

1. Introduction

In order to remain globally competitive, it is critical that organizations establish processes that allow them to adapt to an ever changing market place. This need for agility is particularly pronounced for organizations involved in the design and/or manufacture of complex products. A manufacturing interface that allows design and manufacturing organizations to interact in an effective and efficient manner is essential to realizing the necessary agility. Such an *agile manufacturing interface* must allow designers to quickly asses the level of compatibility between a design and a manufacturing facility, while simultaneously providing manufacturers with the ability to effectively interact with a diverse variety of design organizations.

The Rapid Prototyping of Application Specific Signal Processors (RASSP) Program is a $150M ARPA and US Department of Defense initiative (1994-1997) intended to dramatically improve the way complex embedded digital electronic systems, particularly embedded digital signal processors, are designed, manufactured, upgraded, and supported. The target RASSP improvement is at least a four-fold (4x) reduction in the time to go from design concept to fielded prototype with equivalent improvements in cost and quality. The motivation for the RASSP initiative is the pervasive need for affordable embedded signal processors throughout a wide range of electronic systems.

To achieve the RASSP 4x objective, the RASSP program has supported the development of an agile manufacturing interface.

The *RASSP Manufacturing Interface* provides the mechanisms necessary for diverse design and manufacturing organizations to work synergistically and thereby help ensure first-pass manufacturing success.

In this article, the information sharing and DFx requirements for an agile manufacturing interface are presented. The principle goal of an agile manufacturing interface is to provide the information sharing infrastructure necessary to enable the formation of virtual organizations and to provide them with the robust DFx mechanisms they need in order to develop high quality products in a timely, cost effective manner. The results achieved from the implementation of an agile manufacturing interface to support the RASSP process are presented. These results, realized in a production environment, include a 10x reduction in the time required to transition from design to manufacturing set-up, and a reduction in rework of complex Printed Circuit Assemblies (PCA) of up to 80%. The role that effective information exchange plays in enabling these results is highlighted.

2. Background

A traditional flexible manufacturing system will provide some of the capabilities needed to achieve both the reductions in cycle-time and cost and the improvements in quality sought by the RASSP program. However, stronger coupling between design and manufacturing is needed to fully realize the RASSP objective. To achieve this, an agile manufacturing interface must enable close collaboration between design and manufacturing groups throughout the product development process. An agile manufacturing interface supports this level of collaboration by providing the information sharing infrastructure and DFx mechanisms necessary to develop complex products that achieve *first-pass manufacturing success.*

2.1 Limitations of Traditional Approaches

Today, the information sharing capabilities and DFx mechanisms necessary to support the sophisticated collaboration requirements of agile enterprises do not exist. Typically, little communication occurs between organizations until critical information exchange is required. When an exchange of information occurs, little or no consideration is given to the receiving group's information requirements. This results in what is popularly referred to as the *over-the-wall* paradigm of information exchange.

PCA design and manufacturing organizations all too often operate within this paradigm. The PCA design group will declare a design "finished" once it meets their requirements. Typically, these requirements focus on form, fit, and function while ignoring other considerations such as producibility. It has long been recognized that such a paradigm is inefficient. Companies have taken steps to adopt concurrent engineering principles to ensure that other considerations, such as manufacturability, are taken into account earlier in product design. Several different approaches have been used in an attempt to accomplish this. In the electronics area, one approach has been to incorporate generic Design For Manufacturing (DFM) rules into Electrical CAD (ECAD) systems. ECAD systems equipped with this primitive DFM capability can analyze a PCA and report issues that might negatively impact producibility. A second approach has been to form Integrated Product Development (IPD) Teams consisting of representatives from various disciplines other than design, such as manufacturing. These domain experts are typically collocated with the design team to ensure that their domain's concerns are considered during the design process. Therefore, by including a manufacturing engineer on the IPD team, manufacturing knowledge specific to one facility can be applied to improve the producibility characteristics of product designs.

Both of these approaches have improved the overall situation. However, each has significant drawbacks. Neither approach enables the consideration of multiple manufacturing facilities or production lines, nor do they support the need for agility that is so important in today's defense and commercial environments. In order to be as effective as possible, DFM rules must include knowledge specific to the manufacturing facility and production line that will be used to produce the product being designed. Every manufacturing facility will contain different equipment, will organize its equipment differently, and will have its own idiosyncrasies. These idiosyncrasies can significantly impact producibility and are often poorly documented, if documented at all. Knowledge of this kind is generally distributed among the technicians and manufacturing engineers that operate a manufacturing facility. Because these idiosyncrasies are not considered by the generic DFM checking capabilities that ECAD systems provide, serious producibility issues can exist after the generic DFM rules indicate that no issues remain. Because unknown producibility issues often exist when manufacturing begins, first-pass manufacturing success is rarely achieved.

While the IPD team approach can significantly improve the producibility of designs, it is far from an ideal solution. First, resource limitations constrain the number of manufacturing personnel that can participate on an IPD team, ensuring that only a subset of a manufacturing facility's characteristics can be taken into consideration by the design team. Second, physical collocation of manufacturing experts with designers is expensive and subject to potential interpersonal management issues. Third, the manufacturing knowledge used with this approach exists primarily in the mind of the manufacturing engineer, and thus is volatile from the design organization's point of view. Access to this knowledge can be interrupted or eliminated by factors such as illness, changes in employment, and retirement. Finally, the IPD approach does not easily support the optimization of a design across multiple manufacturing facilities. Because the field of potential manufacturers is severely restricted early in the design process, this approach restricts a design organizations flexibility.

What is needed is an agile manufacturing interface that provides the mechanisms necessary to enable an automated, concurrent engineering environment. Such a solution must eliminate the fundamental, underlying impediments to first-pass manufacturing success of complex products, allow design organizations to quickly interface with different manufacturing facilities, and simultaneously allow manufacturing facilities to effectively interface with many different design organizations [Gad94].

2.2 The ARPA/Tri-Service RASSP Program

The ARPA/Tri-Service Rapid Prototyping of Application Specific Signal Processors (RASSP) program is a 4.5 year, $150M effort aimed at improving the process by which embedded digital electronic systems are developed. The objective of the RASSP program is to reduce by a factor of four the cost and time needed to develop and manufacture embedded signal processing systems while simultaneously improving their quality. RASSP has targeted three areas for development in support of achieving this objective:

- Methodology
- Model Year Architecture
- Infrastructure

The methodology being developed combines concurrent engineering concepts with collaborative teaming approaches. The model year architecture focuses on leveraging commercially available capabilities, coupled with flexible interfaces, to enable regular, low-cost technology upgrades. Improvements in infrastructure are being pursued to increase the effectiveness of the methodology and model year architecture being developed. These infrastructure efforts are focused on two areas. The first is aimed at improving the capability of system level design tools that can be used to automate and improve the decisions made early in the design process. The second focus area is developing improved enterprise integration capabilities, such as enterprise product data management (PDM) systems integrated with workflow management systems and enterprise reuse libraries. By providing integrated workflow management and secure, high bandwidth Internet access, the infrastructure effort will enable application of the methodology and model year architecture across distributed, multi-discipline concurrent engineering teams within a virtual corporation.

A critical component of the enterprise integration capabilities being developed by the RASSP program is the RASSP Manufacturing Interface (RASSP-MI). The goal of the RASSP-MI is to enable first-pass manufacturing success of PCAs within a virtual enterprise by effectively supporting agile manufacturing. This goal directly supports RASSP's goal of significantly improving the quality of and reducing the time and cost required to design and deploy signal processor systems.

3. Agile Manufacturing Interface Requirements

In order to enable the formation of virtual organizations, an agile manufacturing interface must provide a robust information sharing infrastructure coupled with the DFx mechanisms necessary to realize cost effective, first-pass manufacturing success. The following sections discuss these requirements in greater detail and describe how they have been implemented in the RASSP Manufacturing Interface.

The information sharing requirements for an agile manufacturing interface can be divided into two categories. The first requirement is that the information being shared have a*predictable and mutually agreeable form*; that is, its syntax must be understood by both sender and receiver prior to any exchange of information. The second requirement is that the information being shared have a *predictable and mutually agreeable content*; that is, the semantics of the information to be exchanged must be understood by both sender and receiver prior to any exchange of information. This implies that data "flavoring" is not allowed. The content or semantic requirement can be further refined as follows:

Semantic Requirements

- Complete
- Consistent
- Accurate
- Correct

When the data requirements of a receiving activity, such as manufacturing, are not formally understood by a sending activity, such as design, the completeness of the information transferred can not be assured. For example, features of a design considered insignificant to a designer may be crucial to a manufacturer. A design organization may transfer what they consider to be a complete design to the manufacturer, only to discover later that the manufacturer requires more information. Rectifying this situation requires a costly, time consuming iteration between design and manufacturing before production can begin.

The consistency of the information transferred from one activity to another must also be assured. The absence of Integrated Product Data Management (IPDM) between organizations results in an environment in which the consistency of the information may be compromised. For example, it is not uncommon for a manufacturing engineer to make a "minor" change to the layout of a design, such as changing a signal name in the netlist, after the layout has been completed. The resulting lack of consistency between different views of a product can result in costly and unnecessary delays in manufacturing. These delays will result either from problems caused directly by the inconsistent information (when the inconsistency goes unrecognized), or due to the design iterations required to correct the inconsistency.

The most challenging requirements to meet are those of accuracy and semantic correctness. The issue of accuracy arises when information is exchanged in a form other than its native representation. Indeed, even exchanging information in native form can present accuracy problems unless the environment for both sender and receiver (architecture, software environment, etc.) are identical. A design is correct when it meets its functional specification and all "ility" requirements are satisfied, such as

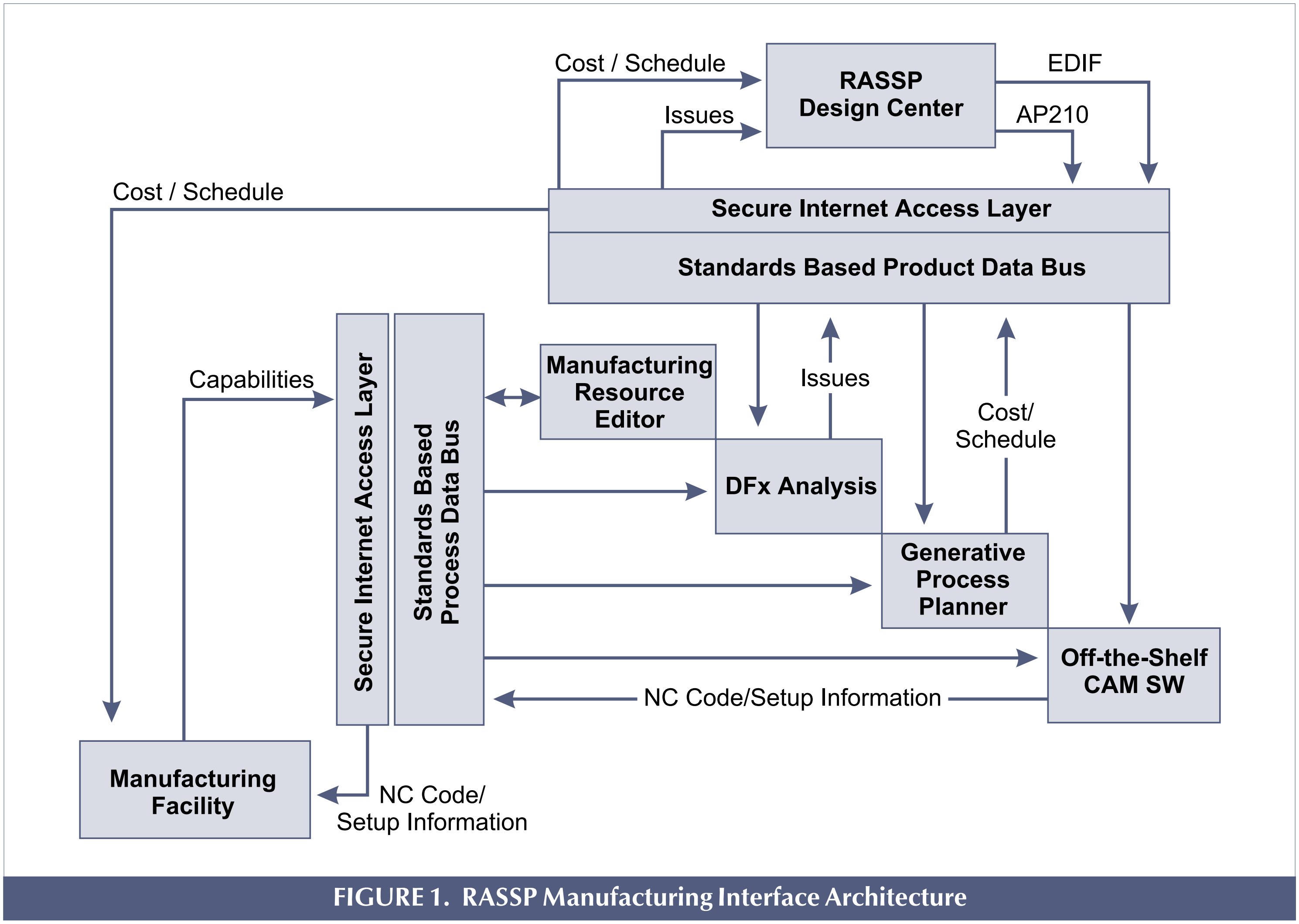

Cost / Schedule
Issues
RASSP Design Center
EDIF
AP210
Cost / Schedule
Secure Internet Access Layer
Standards Based Product Data Bus
Capabilities
Secure Internet Access Layer
Standards Based Process Data Bus
Manufacturing Resource Editor
Issues
Cost/ Schedule
DFx Analysis
Generative Process Planner
Off-the-Shelf CAM SW
NC Code/Setup Information
Manufacturing Facility
NC Code/ Setup Information

FIGURE 1. RASSP Manufacturing Interface Architecture

manufacturability, testability, maintainability, etc. The need to verify that these "ility" requirements are met drives the need for robust DFx checking mechanisms that perform design validation as part of an agile manufacturing interface.

4. The RASSP Manufacturing Interface

Figure 1 presents the RASSP Manufacturing Interface (RASSP-MI) architecture. The role played by each component of the RASSP-MI architecture in realizing the agile manufacturing interface requirements described above is presented in the following sections.

4.1 Information Sharing Standards

The utility of standards in a concurrent engineering environment can be seen in [And94] which describes the role standards played in supporting the concurrent engineering of the engine mount for the Boeing 777 aircraft.

The RASSP-MI makes effective use of robust, widely accepted standards to provide "data buses" which ensure that information can be exchanged between product life-cycle elements (design, manufacturing, testing, field service, etc.) without the loss or duplication of information. The role for information sharing standards in the context of the RASSP agile manufacturing interface is illustrated in Figure 2.

The use of robust standards in the RASSP-MI supports many of the information sharing requirements described previously. Standards such as EDIF [Lau96] and ISO 10303 (STEP) [ISO96] are specified such that they meet the predictable form requirement and can support the content requirements in several ways. By enabling the representation of all needed information, the completeness requirement is supported. The consistency requirement is supported by defining rules as part of the standard that can be used to automatically check the consistency of the information to be exchanged. Lastly, by providing a formal definition of the semantics of the information to be exchanged via an information model, both EDIF and ISO 10303 enable techniques that can ensure the information exchanged is accurate. This is discussed in more detail next.

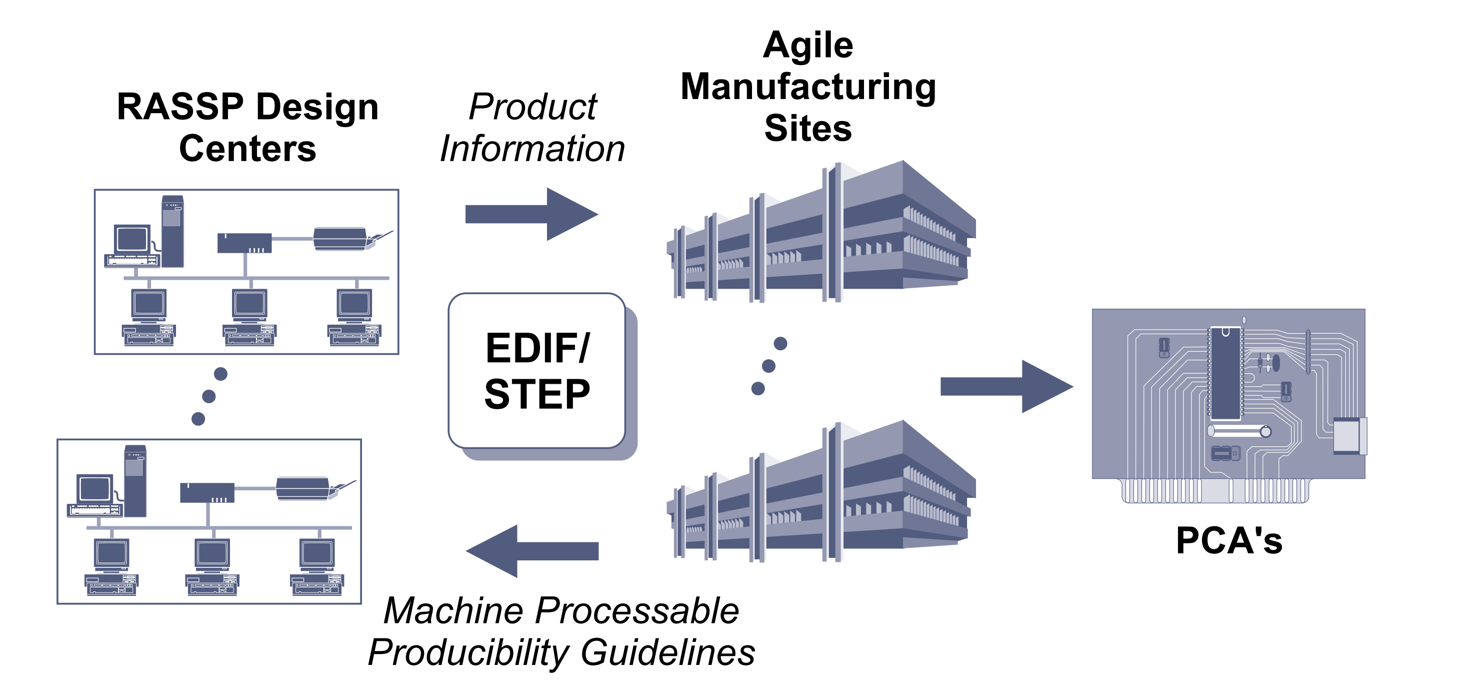
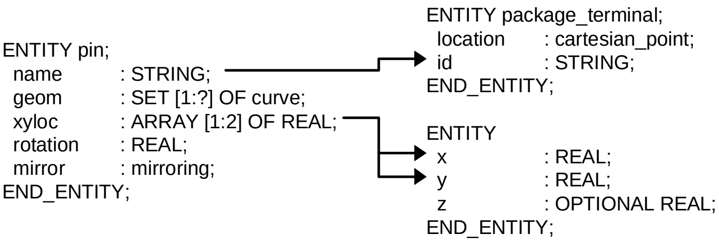

RASSP Design Centers
Product Information
Agile Manufacturing Sites
EDIF/ STEP
PCA's
Machine Processable Producibility Guidelines

FIGURE 2. RASSP Agile Manufacturing Interface

4.2 Assuring Product Data Accuracy

To ensure that the information exchanged accurately conveys the semantics of the original representation, the RASSP-MI makes use of a novel EXPRESS driven approach to data conversion. Using this technique, a formalized definition of the information's semantics is created using the EXPRESS information modeling language [ISO94]. This formal definition enables semantic mappings between different representations to be developed. An example of a semantic mapping is illustrated in Figure 3.



```
ENTITY pin;
  name        : STRING;
  geom        : SET [1:?] OF curve;
  xyloc       : ARRAY [1:2] OF REAL;
  rotation    : REAL;
  mirror      : mirroring;
END_ENTITY;
```

```
ENTITY package_terminal;
  location      : cartesian_point;
  id            : STRING;
END_ENTITY;

ENTITY
  x             : REAL;
  y             : REAL;
  z             : OPTIONAL REAL;
END_ENTITY;
```

FIGURE 3. Semantic Mapping Concept

Note that this mapping is incomplete; a complete mapping of the PIN entity would show the correspondence between all attributes in the two information models. Using the EXPRESS Driven Data Conversion technique defined in [Hin94], these semantic mappings form the basis for accurately converting data from one form into another.

4.3 Assuring Product Data Correctness

As described earlier, the most challenging information sharing requirement, that of semantic correctness, can only be met by robust DFx mechanisms. This drives the need for robust DFx checking mechanisms within the RASSP agile manufacturing interface.

The DFx capability supports the validation of a data-set's correctness relative to specific criteria, such as producibility and testability. The DFx mechanism applies rules to check a data-set against the specified criteria. The values and semantics of the specified criteria exist as a machine processable description of relevant manufacturing capabilities in terms that are meaningful to a designer. An example of a DFx rule is presented below:

Example DFx rule:

$$\forall\ \textbf{traces}\ .\ \textbf{if}\ \ \mathit{trace\_width} < \mathit{min\_trace\_width}$$

$$\Rightarrow \textbf{issue}\ (\ \mathit{min\_trace\_width}\ )$$

In the above example, *trace_width* and *min_trace_width* represent variables. The variable *trace_width* is determined from the description of a product, whereas the variable *min_trace_width* is obtained from the process description that represents a manufacturing facility's capabilities. The DFx analysis mechanism will evaluate the above rule to determine if any trace in the design is narrower than the minimum trace width defined by the manufacturing facility. If a trace is found that violates this condition, an issue is generated that can be resolved through negotiations between design and manufacturing to either alter the manufacturing process, alter the design, or ignore the issue. The manufacturing-facility-specific DFx analysis supports an iterative cycle of design analysis and refinement, which can be repeated until no significant issues remain for a design. While some non-fatal issues may be unresolvable due to design constraints, knowledge of these will result in more realistic manufacturing cost estimates than could be achieved without the aid of such DFx capabilities.

The RASSP-MI provides the necessary DFx capabilities through a World Wide Web (WWW) accessible Producibility Analysis (PA) tool. Together, the WWW and PA support the secure transmission of design information, remote analysis, the secure return of analysis results, and any ensuing negotiations that may be required. An example of the information provided by the RASSP-MI PA tool is presented in Figure 4.

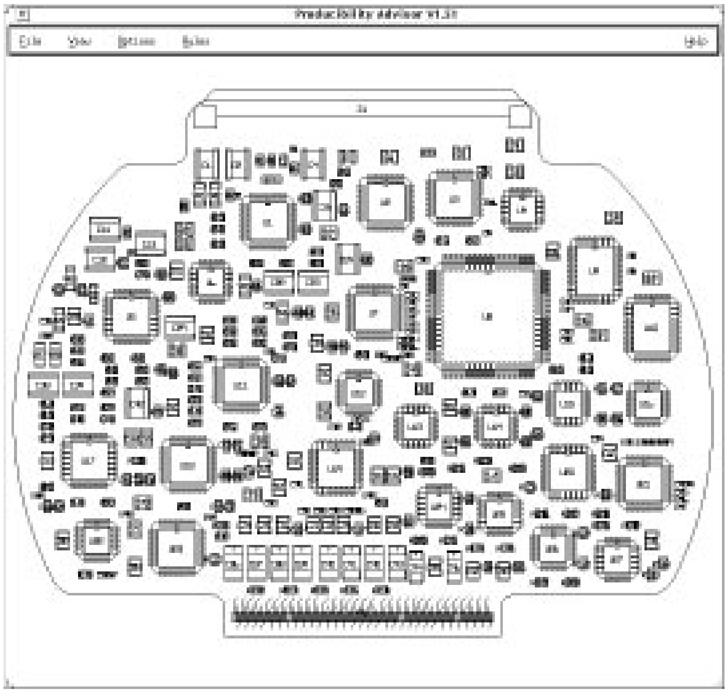


FIGURE 4. View Generated by RASSP-MI Producibility Analysis Tool

The architecture described here has been used to develop the agile RASSP-MI. The results obtained while using the RASSP-MI in a production environment are presented next.

5. Results To Date

The RASSP-MI has been integrated into the RASSP enterprise system and is being utilized by the key PCA manufacturing facility within Lockheed Martin Corporation. Several PCA designs have been processed by the RASSP-MI at this facility to produce a number of PCAs. The results to date indicate a significant reduction in rework and design-to-manufacturing cycle-time. These results are detailed next.

5.1 Baseline Production Environment

Until recently, the process of transitioning PCA product designs from Lockheed Martin's design facilities to its key manufacturing facility involved significant manual data conversion, data reentry, and manual quality assurance procedures. These manual processes required significant time to perform and introduced errors and inaccuracies into data generated for production. These data conversion and quality assurance steps took place after a PCA design was considered "complete" and had been transferred to the manufacturing facility.

Because the manufacturing facility has traditionally not been part of the product design process, manufacturability issues are often present in data received from design. These issues must be resolved before production can begin. Resolution might require a re-design effort by the team originating the design. Because the cost of design modification late in the design cycle is high, manufacturability issues that are not insurmountable are often allowed to remain, even though they increase the recurring manufacturing costs of the product. These problems have not only contributed to difficulty in achieving first-pass manufacturing success, but unnecessarily increased production difficulties and therefore cycle-time and cost.

The RASSP-MI corrects this by facilitating collaboration and negotiation between design and manufacturing engineers throughout the product design process. The role of the RASSP-MI in this process is illustrated in Figure 5.

Two practices are still in place at the manufacturing facility which are legacies of previous product data generation methods. The first of these practices is to manufacture the first three PCAs of the first manufacturing run of a new design by a purely manual process. This is done to ensure that the process of product assembly is well understood. This knowledge can then be used to help address difficulties encountered during automatic assembly of the remaining PCAs. The second legacy practice is to produce only 20 PCAs per manufacturing run (referred to as a*batch*). This is done to provide significant opportunities for manufacturing engineers to fine-tune the automatic assembly process in order to maximize yield.

Prior to use of the RASSP Manufacturing Interface, inaccurate placement of surface-mount components caused significant recurring production difficulty. The manual data exchange process employed did not assure accuracy of placement information to within 1/1000th of an inch. Without this level of accuracy, it was common for small discrete surface-mount components to move during the solder reflow process due to *component drift*. For some components, this movement caused them to make poor or no contact with their designated connection points on the Printed Circuit Board (PCB). Attempts to counter this effect centered around modifying "offset" values in the automatic surface-mount placement equipment. Failures observed during the manufacture of a batch of PCAs would be analyzed by a manufacturing engineer, who would then use the analysis results to modify placement equipment "offset" values in an attempt to correct the component misplacement problem. This approach improved yields, but was never able to eliminate this production problem, even over several years of production of the same design.

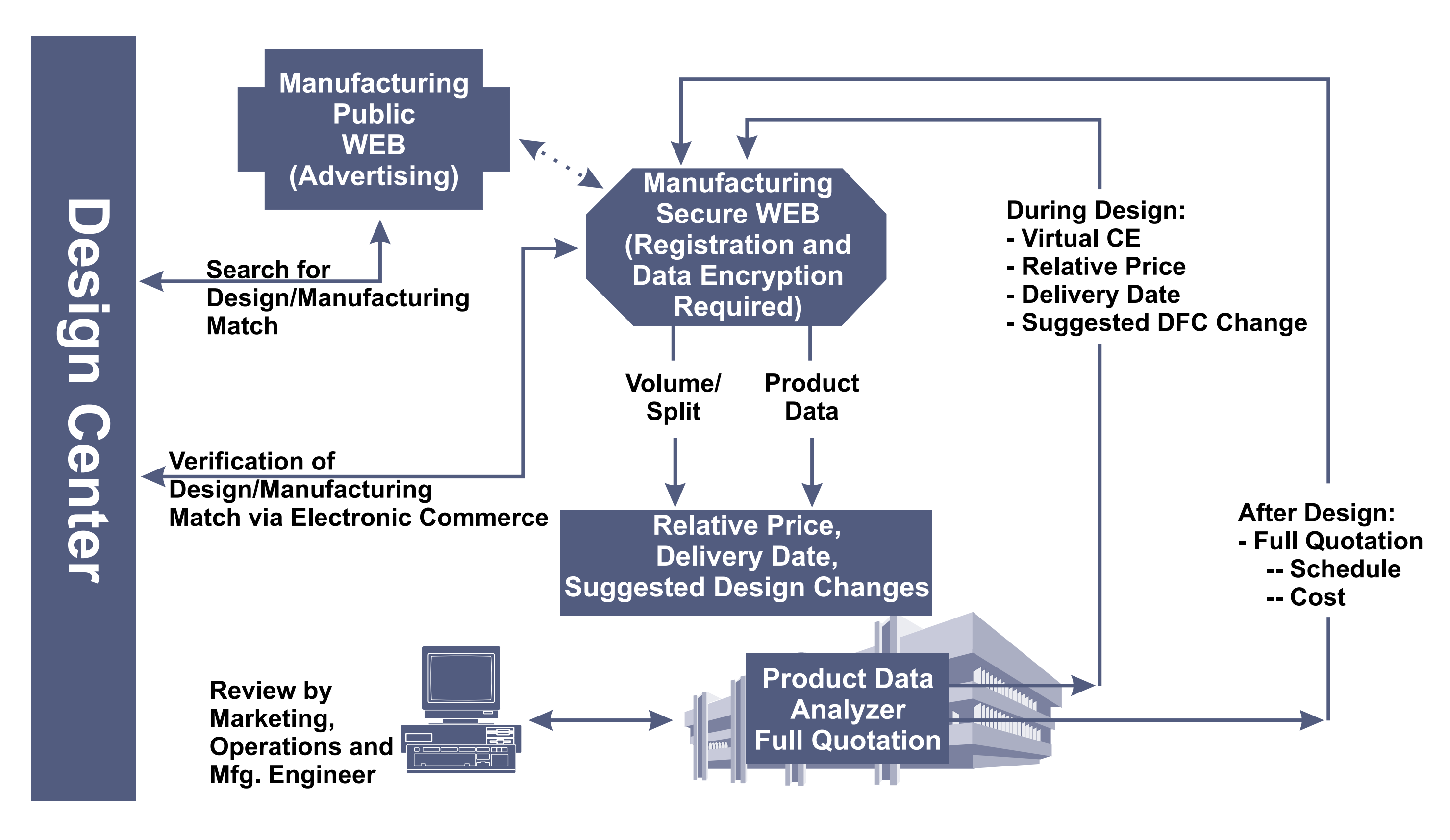

Design Center
Manufacturing Public WEB (Advertising)
Search for Design/Manufacturing Match
Manufacturing Secure WEB (Registration and Data Encryption Required)
During Design:
- Virtual CE
- Relative Price
- Delivery Date
- Suggested DFC Change
Volume/ Split
Product Data
Verification of Design/Manufacturing Match via Electronic Commerce
Relative Price, Delivery Date, Suggested Design Changes
After Design:
- Full Quotation
-- Schedule
-- Cost
Review by Marketing, Operations and Mfg. Engineer
Product Data Analyzer Full Quotation

FIGURE 5. RASSP Agile Manufacturing Interface Methodology

Despite the ingenuity and tenacity of the engineers and technicians supporting this facility, the inaccurate data utilized for production exacted a heavy toll. For one program examined, 100% of 80,000 manufactured PCAs had defects caused by inaccurate placement of surface-mount components. These defects required manual repair. To make matters worse, on average approximately 30% of the components on each PCA required rework. Remarkably, it was determined that the cost required to overcome these difficulties, given the *over-the-wall* paradigm the facility was obligated to operate within, exceeded the cost of performing the repairs.

5.2 Results Using the RASSP-MI

To date, four PCA designs have been processed using the RASSP-MI. These PCA designs are comparable in complexity to the design previously discussed. Using the RASSP-MI, NC code for component placement machines is derived automatically from the original CAD data representation of the design. Therefore, the placement information in the NC code is as accurate as that present in the CAD system. Due to the increased quality of the placement data, it was determined that all of the “offset” values that had been programmed into the surface-mount placement equipment at the manufacturing facility could be reset to 0, which resulted in a simplification of the programming procedures required for this equipment.

Of the four designs processed thus far, three realized first-pass manufacturing success. The remaining design experienced a 70% success rate. For this design, examination showed that a misinterpretation of the manufacturing facility’s information requirements was the cause of the poor yield. The EXPRESS driven approach to data conversion allowed this problem to be quickly identified and corrected.

In addition to supporting first-pass success, the RASSP-MI reduced the design to manufacturing set-up time by more than a factor of 10. The first-pass success and cycle-time improvements were achieved by adhering to the information sharing requirements described previously, eliminating unnecessary process steps, and providing an automated concurrent engineering capability between design and manufacturing.

5.3 Payback Analysis

Equation 1 below defines C_t to be the recurring cost associated with the time required to correct surface-mount component placement errors introduced by the baseline manual data conversion process previously described in section 5.1.

EQUATION 1. Baseline Process Recurring Costs

$$C_t = \left(ME_{lr} \bullet T_m\right) + \left(P_r \bullet T_r \bullet N_p \bullet MT_{lr}\right)$$

Where:

- ME_{lr} is the labor rate of a Manufacturing Engineer
- T_m is the time spent modifying automatic placement “offset” values per day of production
- P_r is the percent of PCAs requiring repair due to poor component placement
- T_r is the average time spent repairing a PCA
- N_p is the total number of PCAs produced
- MT_{lr} is the labor rate of a Manufacturing Technician

Using the RASSP-MI, C_t is negligible. Using the baseline process, C_t was significantly higher. Equation 2 presents the production savings on a per unit basis that has been enabled using the RASSP-MI, S_{RASSP}.

EQUATION 2. Cost Savings Using RASSP-MI

$$[1] \qquad S_{RASSP} = \frac{\left(8 \cdot Years \bullet \frac{250 \cdot Days}{Year} \bullet ME_{lr} \bullet \frac{30 \cdot Man \cdot Minutes}{Day}\right) + \left(80{,}000 \cdot PCAs \bullet MT_{lr} \bullet \frac{15 \cdot Man \cdot Minutes}{PCA}\right)}{80{,}000 \cdot PCAs}$$

$$[2] \qquad S_{RASSP} = \frac{0.5 \cdot Man \cdot Years \bullet ME_{lr} + 10 \cdot Man \cdot Years \bullet MT_{lr}}{80{,}000 \cdot PCAs}$$

$$[3] \qquad S_{RASSP} = \frac{0.5 \cdot Man \cdot Years \bullet \frac{\$200{,}000}{Man \cdot Year} + 10 \cdot Man \cdot Years \bullet \frac{\$150{,}000}{Man \cdot Year}}{80{,}000 \cdot PCAs}$$

$$[4] \qquad S_{RASSP} = \frac{\$20}{PCA}$$

Using Equation 1 and assuming typical fully burdened labor costs, the per unit savings enabled by the RASSP-MI are \$20/PCA, as shown in Equation 2. Given the production rate of the manufacturing facility, the development costs of the RASSP-MI will be paid back in under 6 months.

It should be noted that significant quantities of the four PCA designs processed to date will be produced in the near future. Perhaps even more importantly, given the benefits identified through the use of the RASSP-MI, an additional 25 design projects are expected to be processed by the RASSP-MI over the coming months.

These results highlight the benefits of the agile RASSP Manufacturing Interface and explain why the Lockheed Martin PCA manufacturing facility was identified for a best-practice award [Best95]. With further refinements, it is expected that first-pass manufacturing success of PCAs will be consistently achieved using the capabilities provided by the RASSP agile manufacturing interface.

6 Summary

The goal of an agile manufacturing interface is to enable the formation of virtual organizations by providing the information sharing infrastructure and robust DFx mechanisms those organizations need in order to develop successful products. This paper presented the requirements for an agile manufacturing interface and the results obtained using the agile manufacturing interface developed by the RASSP program (the RASSP-MI) in a production environment. By reducing cost and time-to-market, the RASSP-MI is contributing significantly towards the accomplishment of the RASSP program’s goals of a 4x improvement in cycle-time, quality and cost.

In conclusion, the RASSP Manufacturing Interface allows physically distributed design and manufacturing teams to work collaboratively in a virtual organization to design manufacturability into complex products early in the design process. It also ensures that complex product designs are ready to be manufactured before production begins, thereby ensuring first-pass manufacturing success. For complex products in general, implementations of this capability promise to produce significant reductions in product development time and cost while improving product quality.

Acknowledgments

The research presented in this paper has been supported in part by the Defense Advanced Research Projects Agency (DARPA) Electronics Technology Office (ETO) and the Army Research Laboratory under the RASSP program, subcontract TTM 748358. Special thanks are extended to Ronald A. Pierce and David Dunham (Manufacturing Engineers) who provided invaluable assistance in collecting historical manufacturing facility information and results data.

References

[Gad94] A. J. Gadient, G.R. Graves & J.C. Boudreaux, “PreAmp: A STEP Based Concurrent Engineering Environment for Printed Circuit Assemblies,” In *Proceedings Concurrent Engineering: Research and Applications Conference*, pp. 529-537, August, 1994.

[And94] B. Anderson, S. Ryan, “Using STEP Application Protocols to Enable Concurrent Engineering in Real World Pilot Implementations”, ,” In *Proceedings Concurrent Engineering: Research and Applications Conference*, pp. 349-353, August, 1994.

[Lau96] Lau, R.Y.W., EDIF: Electronic Design Interchange Format Version 4 0 0 Information Model, Electronic

Industries Association, EDIF Steering Committee, 1996.

[ISO96] ISO/DIS 10303-210:1996, Industrial automation systems and integration ¾ Product data representation and exchange ¾ Part 210 Printed circuit assembly product design data.

[ISO94] ISO 10303-11:1994, Industrial automation systems and integration ¾ Product data representation and exchange ¾ Part 11: Description methods: The EXPRESS language reference manual.

[Hin94] L.E. Hines, A. J. Gadient, "EXPRESS Driven Data Conversion," In *Proceedings Concurrent Engineering: Research and Applications Conference*, p. 313-322, August, 1994.

[Best95] "Report of Survey Conducted at Lockheed Martin Electronics & Missiles, Orlando, FL", Best Manufacturing Practices Center of Excellence, College Park, Maryland, April 1995.

Anthony J. Gadient, Lynwood E. Hines
Advanced Technology Group, SCRA
5300 International Blvd.
N. Charleston, SC 29418
gadient@scra.org, hines@scra.org

John Welsh
Lockheed Martin Advanced Technology Laboratories
Camden, NJ
jwelsh@atl.lmco.com

Andrew P. Schwalb
Lockheed Martin Corporation
498 Oak Road
MP-A22
Ocala, FL 34472
aschwalb@ocala1.lmc-ocala.com

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • ADL

ARPA/VI VHDL Educator's Workshop

November 10-12, 1996 (Tentative)

Silicon Valley, CA

This course is designed for educators interested in introducing VHDL into their undergraduate and graduate curriculum. An introduction to the language and material for courses, including notes, labs, test problems and answers will be provided.

Course topics will include:

- Basic VHDL
- Behavioral VHDL
- Structural VHDL
- System Level VHDL

For more information contact:

Professor James Aylor
(803) 760-3376
http://rassp.scra.org/VHDL.WORKSHOP

RASSP Steering Committee

ARPA (ETO)		
	Randy Harr	**Program Manager**
ARMY		
	Randy Reitmeyer	**Administrative COTR, Lockheed Martin - Advanced Technology Laboratories**
	Arne Bard	**Technical COTR, Lockheed Martin - Advanced Technology Laboratories**
NAVY		
	Ingham Mack (ONR)	
	Gerry Borsuk (NRL)	
	J. P. Letellier (NRL)	**Administrative COTR, Lockheed Martin - Sanders**
		Technical COTR, Lockheed Martin - Sanders
AIR FORCE		
	Stan Wagner	**Educator Facilitator and Technology Base**
	John Hines	**COTRs**

RASSP Digest-Rapid Prototyping of Application Specific Signal Processors

The RASSP Digest is published quarterly and provides information for and about the RASSP Program and rapid systems development. For more information, contact Dr. Anthony Gadient or Dr. Vijay Madisetti, Editors, at the addresses below:

Anthony J. Gadient
Phone: 803-760-4082
FAX: 803-760-3349
Email: gadient@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

Vijay K. Madisetti
Phone: 404-853-9830
FAX: 404-853-9171
Email: vkm@ee.gatech.edu
Georgia Tech
School of Elec. & Computer Eng.
Atlanta, GA 30332-0250

Bryan R. Bryant
Managing Editor
Phone 803-760-3363
Email: bryant@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

Calendar of Events

CERA **CE96** For more information: http://ce-toolkit.crd.ge.com/ce/master.html	**August 26-28, 1996**	**University of Toronto,** **Toronto, Canada**
1996 IEEE Digital Signal Processing Workshop For more information: dspws96@tele.unit.no	**September 1-4, 1996**	**Alexandra Hotel,** **Loen, Norway**
ICSPAT - DSP World Expo For more information: dsp@mfi.com	**October 7-10, 1996**	**Boston, MA**
1996 International Test Conference **"Test and Design Validity"** For more information: itccon@aol.com	**October 20-24, 1996**	**Sheraton Washington Hotel** **Washington, DC**
VHDL International Users' Forum (VIUF) **Fall 1996 Conference & Exposition** For more information: erol@ee.duke.edu	**October 27-30, 1996**	**OMNI Hotel** **Durham, NC**
ARPA/VI VHDL Educator's Workshop For more information: info@rassp.scra.org http://rassp.scra.org/VHDL.WORKSHOP/	**November 10-12, 1996** (Tentative, check WWW for details)	**Silicon Valley, CA**
1996 International Conference **on Computer-Aided Design (ICCAD)** For more information: icpubpap@dac.com	**November 10-14, 1996**	**San Jose, CA**

For additional RASSP information including an electronic version of this publication, visit the RASSP World Wide Web site at

http://rassp.scra.org

Recognized by IEEE as one of the top three WWW sites on DSP.

IEEE Spectrum, January, 1996

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • ADL

SCRA
5300 International Blvd.
N. Charleston, SC 29418

September
1996

RASSP Digest

RASSP - Rapid Prototyping of Application Specific Signal Processors

Vol. 3, September, 1996

Technology Base Efforts

Methodology

RASSP
Reinventing
Electronic
Design

Architecture Infrastructure

DARPA • Tri-Service

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • ADL

In This Issue

Editor's Corner

RASSP ESDA Tools Part 1: Detailed HW/SW Codesign

RASSP ESDA Tools 2: Front-End HW/SW Codesign Design Tools

RASSP Model Libraries: VHDL

RASSP and VHDL Issues

RASSP Digest Theme: Technology Base Efforts

Vijay K. Madisetti and Anthony J. Gadient

RASSP is a $150 million DARPA research program headed (in separate efforts) by Lockheed Martin's Advanced Technology Labs and Sanders Corporation. In addition to these prime efforts, over two dozen contractors (including universities, non-profit and commercial organizations) contribute to RASSP technology as part of several Technology Base contracts. Many impressive results from these efforts are documented in this special issue of the *RASSP Digest*.

The typical implementation of a RASSP system consists of three stages:

(1) hardware design and integration,

(2) software integration, and

(3) hardware-software integration and test.

These are described as follows:

(1) Hardware design and integration involves design of the architecture of the multiboard system (processors, interconnect, and topology), building and configuring the runtime deployed platform, designing and installing cabling, configuring each module, and assigning interrupts and memory addresses for each hardware subsystem. The goal is to create a memory map of the entire system and also deal with packaging and test issues.

(2) Software integration (primarily control and diagnostic software) involves developing device drivers and I/O interface libraries to enable communication with the application software. It also includes functional and unit testing of the runtime utility and software modules, and testing the various I/O utilities independent of the application software.

(3) Hardware-software integration and test involves designers using external test equipment and software to stimulate the prototype in an environment similar to the target one. Designers also provide an application development that allows the user to map their application onto the runtime hardware-software platform completed in stages (1) and (2). In a typical RASSP system, the second stage usually requires development of about twenty times the software (in lines of uncommented code) as the third stage.

RASSP is investigating two rapid prototyping approaches. In the first approach, an application is mapped onto a predefined, off-the-shelf embedded platform through code generation. In the second approach, the platform itself is designed, and integrated together with the application software.

In the first section of this issue of the *RASSP Digest*, the RASSP Technology Base results using the first approach described are presented. In these efforts, the runtime hardware and software is already pre-designed and available commercially off-the-shelf (COTS). In both articles, the Mercury Raceway platform is the target platform onto which the application is rapidly ported. Since the first two stages of the design process described previously are eliminated, this environment allows for rapid implementation as the two articles in this section show.

In the second section of the *Digest*, we focus on the second approach, where the actual hardware and the software architecture themselves are being designed. Clearly, there is a greater freedom in design choices with this approach, and more effort required in the design, integration and test. The articles in this section describe the new methodologies and tools developed by the RASSP Technology Base to address the challenges this increased flexibility provides.

Virtual prototyping in RASSP depends on the availability of a rich set of verified libraries at multiple levels of abstraction to aid in the design and verification process. The third section of the *Digest* describes the development of libraries from the performance level to the component level. Techniques for automating model generation and verification, and the issues of hybrid modeling are discussed.

RASSP and VHDL have been closely linked due to the very powerful expressive features of the language. Extensions to VHDL, as developed by some Technology Base efforts, are presented in the final section of the *Digest*.

We hope you will find the new technology and tools presented in this special issue of significant utility in your quest for an efficient system-level design automation environment. Extensive details on these RASSP developments are also found on the RASSP WWW server (*http://rassp.scra.org*).

Vijay K. Madisetti
ECE,
Georgia Tech.
Atlanta, GA 30332-0250
vkm@ee.gatech.edu

Anthony J. Gadient
SCRA
5300 International Blvd.
N. Charleston, SC 29418
gadient@scra.org

An Application Configuration Language for Multicomputer Tool Development

Barry Isenstein, Michael Krueger and Arlan Pool

Abstract

Developing software applications for scalable, heterogeneous platforms is a highly specialized, time consuming task. This paper discusses a new development that provides an interface for developers and is an intermediary for innovative application development tools. A 'component software' approach insulates both the application developer and tool developer from platform issues while facilitating high-performance execution of applications. Central to the development is a new open application framework that uses an application configuration language based on the well-known Tool Command Language (Tcl), written by John K. Ousterhout of Sun Microsystems and UC Berkeley. Adoption of the framework by advanced tools potentially offers a dramatic programming productivity gain over existing practices.

1. An Application Framework

An open application framework provides a pre-existing application structure to assist the developer in creating portable applications. The application framework described herein should be readily understood by RASSP developers since it provides a software development methodology analogous to the methodology applied in computer aided engineering (CAE) tools for developing complex chip or board designs. Hardware designers approach a complex design as a set of interconnected components as opposed to a monolithic entity. Users of CAE tools have various text-based and visual-based methods for representing a design. Early and often modeling and simulation of a design is essential for development. Portability and reuse are inherent in the CAE tool flow and further amplified by the establishment of standards.

The application framework addresses the requirements of software engineers who implement single-program multiple-data (SPMD) and multiple-program multiple-data (MPMD) scalable systems. For example, the framework is applicable to digital signal processing (DSP), image processing, simulation, or any application modeled as series of data transformations or as event-driven processes.

2. Application Configuration Language

Central to this development is the definition of a robust scripting language for expressing the relationships between the software application and a heterogeneous processor configuration called the Application Configuration Language (ACL). ACL is implemented as an extension of Tcl. Tcl is a procedural language that provides a complete set of control statements (e.g., lists, arrays, variables, procedures).

Talaris is a project name at Mercury Computer Systems for an ACL application framework for developers and high-level tools for RACE multicomputers. ACL, however, is system independent, and other ACL frameworks for different targets are underway. The points about the Talaris Framework will apply to other ACL frameworks.

The Talaris Framework uses component software concepts to provide a software reuse model and an application building mechanism that replaces the use of cumbersome *make* and *shell* scripts. These are the Talaris Framework characteristics:

- Centralizes hardware and software configuration information.
- Expresses assignment, data flow, and scale information algorithmically in a rich and natural manner.
- Supports scaling of heterogeneous system components without imposing an application design model.
- Remains independent of system-specific APIs and supports legacy executable programs.
- Eliminates all target-specific setup and initialization code.
- Enables fast turnaround of configuration changes.
- Supports deployment.
- Creates open framework interfaces that leverage standards.

Talaris reduces application building efforts by providing a ACL script and an inventory of software Programs and Modules. Software Programs are self-contained executable files (e.g., image files) whose interaction is opaque to the Framework. Software Modules are functions or subprograms that have Ports. Each module will run as its own thread. Ports represent how a Module sends messages, shares memory, transfers data, or synchronizes with other Modules. ACL does not impose any particular Port API and can use generic mechanisms.

3. Framework Structure

Figure 1 shows the Talaris Framework. The Generator receives input from three sources: 1) ACL commands, 2) reusable Modules,

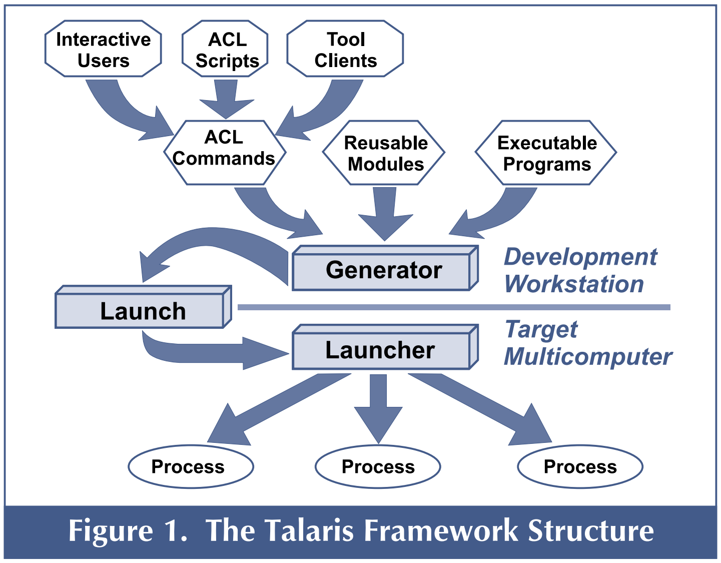


Figure 1. The Talaris Framework Structure

and 3) executable Programs. ACL commands may be user interactive, recorded in a user-written script, or produced from a tool client. ACL describes the hardware configuration and the software application. For example, ACL describes how Modules connect to each other. In the CAE analogy, a Module is comparable to an integrated circuit (IC), a Port to a pin on that IC, and the connection of Ports to a wire that connects pins.

The Generator, which runs on a workstation, creates a Launch Kit. By interpreting the ACL commands, the Generator builds executable images from Modules as needed. The Generator also places run-time setup information into each Launch Kit.

The Launcher, which runs on a designated processor, analyzes the Launch Kit. The Launcher then loads images, sets up the global interprocessor communication environment, and spawns each process. The application is now running. Embedded applications require only a Launch Kit(s) and the Launcher.

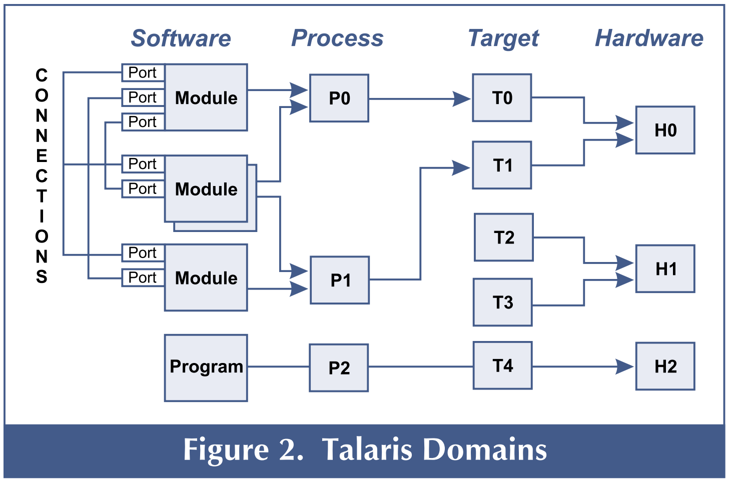


Figure 2. Talaris Domains

Figure 2 introduces the concept of Talaris Domains. Domains are containers for components and are useful for dealing with heterogeneity, application scale, and configuration. It is desirable that changing any one of these aspects result in only a minimal effort. Connections take place within a Domain. Assignments are across Domains.

The four Domains are as follows:

Software Domain

The Software Domain is where instances of Modules and their Port connections are created. Heterogeneity is supported by specifying that Modules are either written in a portable language or for performance reasons, critical Modules have an optimized version for each processor type. In a modeling environment, Modules are behavior models of the function instead of functional components. The Software Domain represents the functionality of the application.

Process Domain

The Process Domain depicts the assignment of Modules to processes. Process scheduling policies are parameterized in this Domain. The Process Domain places an executable software structure on the functional application.

Target Domain

The Target Domain describes the ideal assignment of processes to Compute Environments (CEs). The Target Domain expresses the ideal scalability assignment of an application that might prove useful for actual hardware assignment.

Hardware Domain

The Hardware Domain is the processor assignment that represents the actual hardware present. Connections of components within the Hardware Domain (not shown in Figure 2) represent the processor connection topology. In a modeling environment, the Hardware Domain represents simulation models of existing or future hardware.

4. More on ACL

ACL includes all standard Tcl commands. A partial list of Tcl commands illustrates usefulness for implementing embedded command processing:

variables: set, $x, array, $x(y), incr, argv, env

control: if, for, foreach, while, exit

lists: list, lappend, llength, lindex, lreplace, concat

strings: string, join, split, append, format

functions: proc, return

I/O: open, close, eof, gets, puts, cd

other: catch, eval, exec, expr, trace

Following is a list of the ACL commands:

types: declare, get_type, delete

instances: create, delete

assignments: assign, deassign

connections: connect, disconnect

properties: set_property, get_property, delete_property

scale: set_scale, get_scale

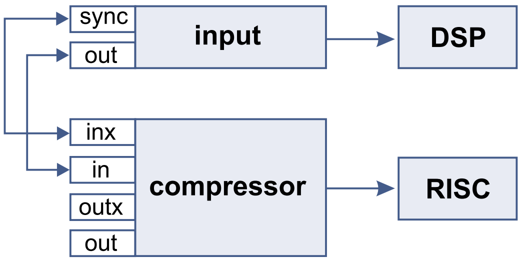
run-time environment: target

information: query

application construction: generate

application control: load, initialize, start, run

It is not within the scope of this paper to explain detailed ACL examples. We refer the reader to the ACL tutorial and other documents available at http://www.mc.com/technology.html.



```
declare Module PulseInput {Sem sync Msg-q out}
declare Module PulseCompressor {Sem inx
Msg-q in Sem outx Msg-q out}
create PulseInput Input
create PulseCompressor compressor
connect input.sync compressor.inx
connect input.out compressor.in
assign input -to DSP
assign compressor -to RISC
```

Figure 3. A Simple ACL Example

Figure 3 shows an illustrative example of a simple Software Domain connection to a Hardware Domain assignment (explicit Process and Target Domain assignments are optional).

5. ACL Development Scenario

An ACL development scenario starts with an inventory of Modules and Programs. Modules are supplied as libraries by vendors, created with code generation tools, or manually written. This inventory provides a reusable code base for concurrent and future projects.

An ACL script is required that 1) defines and creates Modules and Programs, 2) defines and creates Processes (optional), 3) describes the Target (optional) and Hardware Domains, and 4) makes the assignments between Domains. The ACL script could represent a manual effort, an automatic task from a high-level tool, or a combination of manual and automatic efforts.

The ACL script is then loaded into the Generator. The Generator incrementally builds a model of the application and database of connections and assignments. The Generator is eventually given the ACL command to "generate," and the following two step process takes place:

Step 1: The Generator analyzes the application

- determines what runs where
- devises a plan for initializing connections
- performs additional validation
- links Modules and Agent Module to form generated executables
- creates the Launch Script used later to initialize and start the application

Step 2: The Generator creates the Launch Kit with

- Launch Script (loading, spawning, IPC setup)
- executables for generated programs (made from Modules)
- executables for user programs (Programs)

The single ACL command "run" instructs the Launcher to do its job. In summary form, the Launcher

- analyzes the Launch Kit
- extracts and executes the launch script
- loads executables
- sets up global IPCs
- spawns threads in processes

To accomplish its function, the Launcher uses a special Module provided by Talaris called the Agent Module. The Agent Module is linked into generated programs and provides the "main()" entry point. The Agent Module uses data in the Launch Kit provided by the Launcher to create and initialize local IPCs and prepare Modules for execution. The Agent Module then reports back to Launcher, and when directed by the Launcher, starts the Modules.

6. Comparision to Conventional Development

Table 1 illustrates the comparison of a conventional software development cycle to using the Talaris Framework. As indicated in Table 1, ACL, the Generator, and the Launcher replace, or obviate, many of the conventional development tasks.

Although Talaris supports legacy code through the Program entity, achieving the main benefits of the Framework occur when employing Modules (boxed area in Table 1). The Framework allows the code developer to concentrate on application code rather than learning platform-specific API protocols. Application development can then be focused on functional correctness (i.e., making connections in the Software Domain), scheduling policies (parameterized in ACL Software and Process Domains), and on application mapping (i.e., making assignments across Domains).

Connections and assignments can be expressed algorithmically using powerful Tcl and ACL constructs. For example, it is possible to create a process-to-processor mapping algorithm in ACL that responds to parameter changes in hardware interconnect, scale, and heterogeneity.

The developer determines the optimum assignment and mapping, either by an algorithmic or a manual approach. The Talaris Framework does not impact application performance since no generated code is added to the application. Modules are directly connected as if they were coded manually. The application performance will depend on the effectiveness of the assignment and scheduling, and the platform's implementation of the Port APIs.

CONVENTIONAL DEVELOPMENT	TALARIS APPLICATION FRAMEWORK	
create/port application-specific code	SAME	developers focus on the application
create makefile for compilation	SAME	Talaris accepts compiled code
learn system-specific APIs	n/a	Modules can use generic mechanisms
create/debug setup code	ACL	developer specifies connections
add to makefile for compilation	n/a	setup is data-driven
create makefile for building executables	ACL	developer specifies assignments
build and organize executables	n/a	performed by Generator
create/debug run-time scripts/code	n/a	internal to Launcher
debug and tuning	SAME	platform-specific tools

Table 1. Comparison of Development Tasks Between Existing Practices and the Talaris Framework

7. Talaris as a Tool Intermediary

The Talaris Generator presents itself as an interactive shell on the development system. A Talaris project goal is to facilitate rapid third-party development of advanced tools by eliminating many of the platform-specific build and run issues that are typically a large portion of a tool porting effort.

Advanced tools will “hide” the ACL framework from the application developer. Software Modules become reusable components for quick and easy manipulation by the developer using innovative tools with clever user interfaces. A tool presents a helpful interface(s) (e.g., data-flow GUI) to the programmer for connecting and assigning software and hardware Modules. Code generation tools could be invoked to create software Modules. Intelligent tools may assist the developer in offering semi-automated hinting to complete automation of Module assignment to hardware elements. The tool appropriately issues ACL commands to a framework and when directed, the framework dutifully constructs and runs the resulting application.

With an ACL framework ported to other platforms, a single tool can offer portable execution by simply directing output to a different target system.

8. Future Work

Work with Talaris Framework will focus on three areas: 1) robustness and standardization, 2) high-level tool development, and 3) research for large-scale systems.

Talaris represents a new approach which has been a missing ingredient in the effort to develop scalable systems. With sufficient experience by both application developers and tools vendors, formal standardization of ACL will be pursued so that control of the interface will be passed to a public body.

Mercury will assist vendors and research organizations interested in interfacing existing high-level GUI-based tools to the Talaris Framework. Mercury expects various types of tools to be available in the areas of performance modeling and application building, in addition to porting the Talaris Framework to other hardware platforms.

The most promising aspect of the framework is the potential to solve difficult large-scale system problems. Topic areas include automated scheduling, assignment, dynamic reconfiguration, and fault resilience.

Barry Isenstein, Michael Krueger and Arlan Pool
Mercury Computer Systems, Inc.
199 Riverneck Road
Chelmsford, Mass. 01824
barry_isenstein@mc.com

Autocoding Update

Christopher B. Robbins

Abstract

Management Communications and Control, Inc. (MCCI) is developing autocoding tools under subcontract to Lockheed Martin, Advanced Technology Laboratories, Camden (LM/ATL). Under the technology base program, MCCI has developed the Graph Translation Tool (GrTT). These tools are intended to complement each other and will be integrated in MCCI's autocoding toolset. The alpha version of our toolset has recently been evaluated by comparing the autocoded RASSP synthetic aperture RADAR (SAR) benchmark with an earlier hand optimized implementation. MCCI has recently completed work under its RASSP tech base contract on the Graph Translation Tool (GrTT). In technical testing, we used the same SAR benchmark. This article reviews the autocoding process and describes the combined use of GrTT and the autocoding toolset. A summary of the LM/ATL evaluation and MCCI's GrTT testing is presented.

1. Autocoding Process

The autocoding process translates the data flow graph software architecture specification to an efficient parallel application for the target hardware architecture. Inputs to the autocoding process are application specifications in the form of a Processing Graph Method (PGM) data flow graph and hardware architecture specifications. Application graphs are target independent. This is an open API, specifying applications in a form that may be automatically coded for all RASSP targets. PGM is a target independent specification method. Target independent domain primitives are specified as the node executables. The autocoding process is applied in two levels; top level design and detailed design and coding.

In top level design, the Equivalent Application Generator Tool is used to partition input graphs into a connected set of component graphs specifying execution and functional behavior that is identical to the original graph. If allocations are not already specified, application graphs may be partitioned into software allocation and hardware partition graphs. The tool is then used to generate PGM graphs for each hardware partition and the entire software allocation. The software allocation graph is then repartitioned to the programmable elements of the target architecture. An equivalent application graph for the software allocation is created. In the equivalent application graph, each equivalent node replaces a software partition. At the ports of the equivalent nodes, the equivalent application graph behavior will be identical to that of its respective partition graph. The equivalent application graph and the set of hardware and software partition graphs complete the top level software specification. A performance assessment of the equivalent application graph is generated. The performance assessment may be used to quickly reject unfeasible designs.

GrTT is used in top level design to validate requirements capture from the algorithm development stage of codesign. GrTT accepts a partition graph with enumerated sets of external graph controls as its input. GrTT generates Ada graph behavior models that exhibit input graph behavior for all enumerated values of control. Behavior consists of a procedure implementing a sequence of domain primitive executions and intermediate queue states. A GrTT test support utility is provided that executes GrTT procedures as single node applications. GrTT behavior models are executed on test vectors generated by higher level design tools to verify requirements capture by the top level design. This feature may be used to verify both software and hardware partitions. Because of the closeness of Ada and VHDL syntax, GrTT behavior models may be reused as VHDL behavior architectures for hardware partitions.

The second level of the autocoding process is a generation of source code for the partition executables and a configuration file for the equivalent application graph. Configuration files are target architecture specific application descriptions binding graphical equivalent application entities to specific target hardware realizations. A load image specification is then created specifying the complete application to the compiler supporting the target architecture. Unit testing of executables is supported at this level.

Using the MPID Generator tool, each partition graph is translated into 'C' source code for an executable program implementing the partition behavior. MPIDs (Multi Processor Interface Descriptions) are optimized implementations of partition behavior utilizing the math library primitives supported by the target processor. MPIDs become the primitives of the equivalent nodes. Test images of the MPIDs are created. These are single node applications used in validation testing of what are in effect the application's CSUs. Unit testing results are compared with GrTT test vectors to validate partition translation. With its executables validated, correct execution of the application can be expected.

The Application Generator generates configuration files from the equivalent application graph and the hardware description file. Hardware description files are automatically built from the input architecture description and may subsequently be edited by the user. A run-time support (RTS) utility is provided with the tools. The RTS provides application management, execution, and external control interface support. When instantiated, executable and controllable images of the application are created by the RTS from the application's configuration file. A load image specification consisting of all configuration files in the application system, all supporting MPID source files, and necessary system files is then automatically generated.

2. Autocoding the SAR Benchmark

Engineers at LM/ATL Camden recently completed an evaluation of the alpha version of our autocoding tools. In this evaluation, the RASSP SAR benchmark was implemented using our toolset. The work required to develop the application and the code performance was compared with that of the previously hand coded effort. Demonstration cases were developed with GrTT for inclusion in

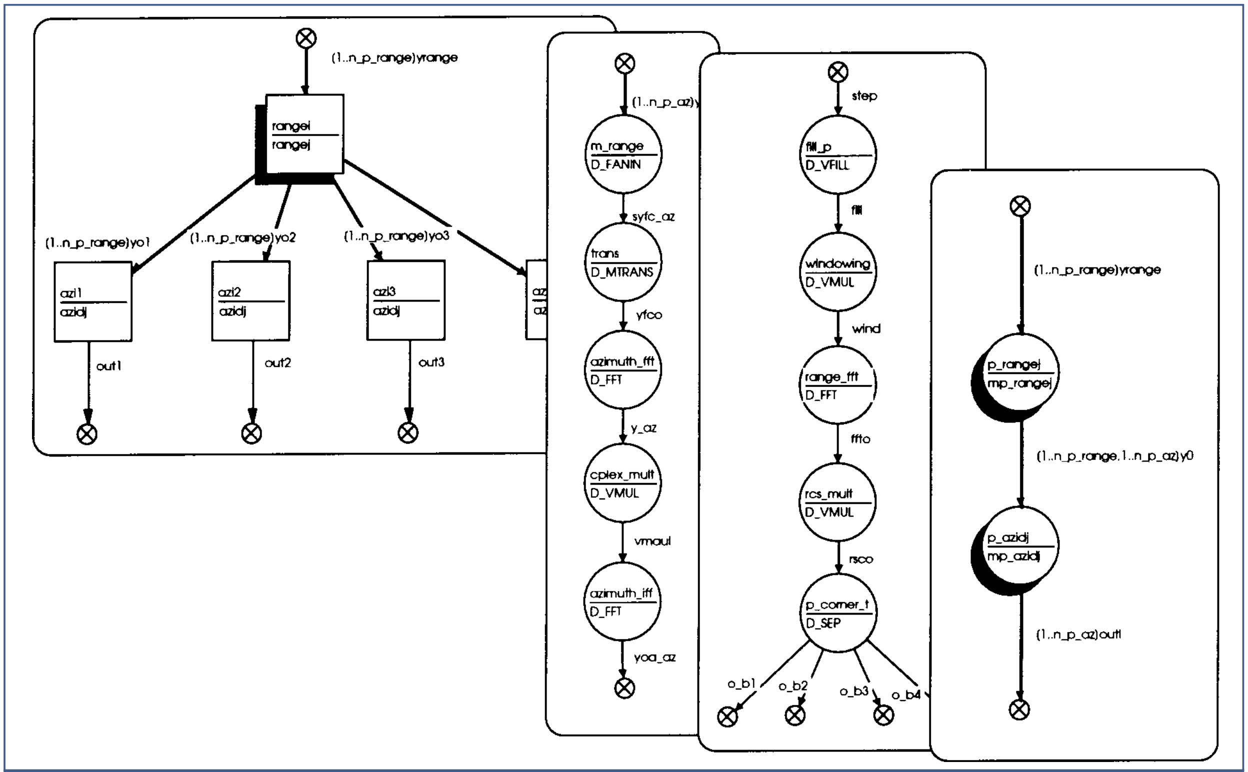


Figure 1. SAR Benchmark Software Allocation Graph, Range and Azimuth Partition Graphs, and Equivalent Application Graph

the technical report for our tech base effort. We used the same benchmark to demonstrate use of GrTT in the context of an application development scenario.

Since the alpha version of our tools does not include the Equivalent Application Generator, partition graphs were manually created from the input software allocation graph. Figure 1 shows the allocation graph. It includes range processing and azimuth processing subgraphs. The range processing and azimuth processing partition graphs and equivalent application graph created from the allocation graph are also shown.

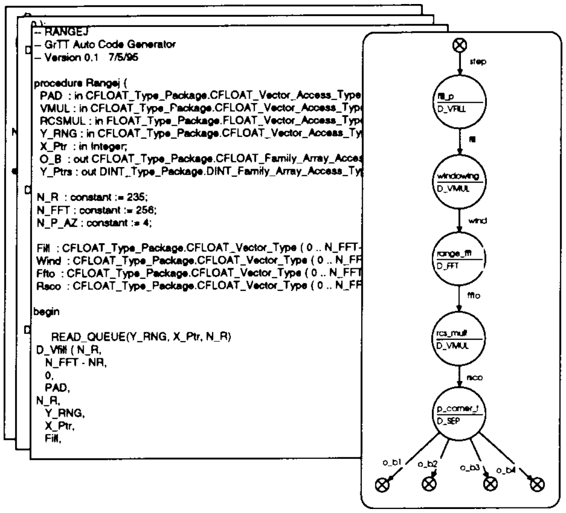

```
-- RANGEJ
-- GrTT Auto Code Generator
-- Version 0.1   7/5/95

procedure Rangej (
  PAD  : in CFLOAT_Type_Package.CFLOAT_Vector_Access_Type
  VMUL : in CFLOAT_Type_Package.CFLOAT_Vector_Access_Typ
  RCSMUL : in FLOAT_Type_Package.FLOAT_Vector_Access_Typ
  Y_RNG : in CFLOAT_Type_Package.CFLOAT_Vector_Access_Ty
  X_Ptr  : in Integer;
  O_B  : out CFLOAT_Type_Package.CFLOAT_Family_Array_Acces
  Y_Ptrs : out DINT_Type_Package.DINT_Family_Array_Access_Ty

 N_R  : constant := 235;
 N_FFT : constant := 256;
 N_P_AZ : constant := 4;

 Fill  : CFLOAT_Type_Package.CFLOAT_Vector_Type ( 0 .. N_FFT-
 Wind  : CFLOAT_Type_Package.CFLOAT_Vector_Type ( 0 .. N_FF
 Ffto  : CFLOAT_Type_Package.CFLOAT_Vector_Type ( 0 .. N_FFT
 Rsco  : CFLOAT_Type_Package.CFLOAT_Vector_Type ( 0 .. N_FF

begin

    READ_QUEUE(Y_RNG, X_Ptr, N_R)
 D_Vfill ( N_R,
   N_FFT - NR,
   0,
   PAD,
 N_R,
   Y_RNG,
   X_Ptr,
   Fill,
```

Figure 2. Autocoded Range Partition Ada Behavior Model

GrTT behavior models were generated from the range partition graph to demonstrate GrTT in the context of the benchmarked development scenario. Figure 2 shows an excerpt from the GrTT behavior for the range partition. MPID source files were generated for the partition graphs using the MPID Generator tool. Figure 3 shows an excerpt of the MPID generated from the same range processing partition graph. The GrTT behavior model was

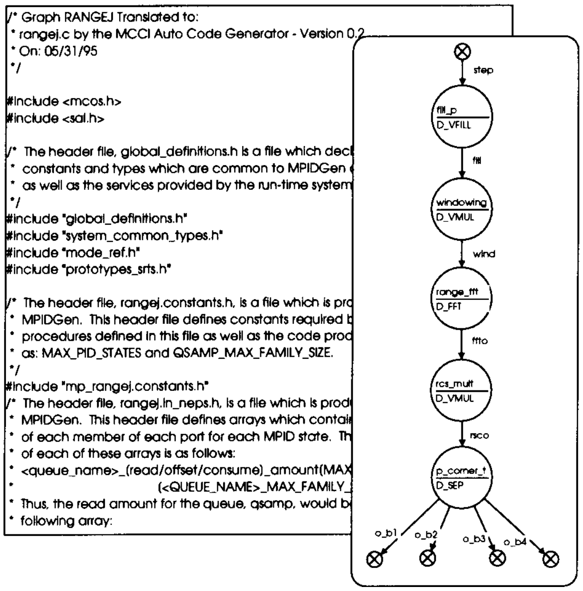

```
/* Graph RANGEJ Translated to:
 * rangej.c by the MCCI Auto Code Generator - Version 0.2
 * On: 05/31/95
 */

#include <mcos.h>
#include <sal.h>

/*  The header file, global_definitions.h is a file which decl
 *  constants and types which are common to MPIDGen
 *  as well as the services provided by the run-time system
 */
#include "global_definitions.h"
#include "system_common_types.h"
#include "mode_ref.h"
#include "prototypes_srts.h"

/*  The header file, rangej.constants.h, is a file which is pro
 *  MPIDGen.  This header file defines constants required b
 *  procedures defined in this file as well as the code prod
 *  as: MAX_PID_STATES and QSAMP_MAX_FAMILY_SIZE.
 */
#include "mp_rangej.constants.h"
/*  The header file, rangej.in_neps.h, is a file which is prod
 *  MPIDGen.  This header file defines arrays which contai
 *  of each member of each port for each MPID state.  Th
 *  of each of these arrays is as follows:
 *  <queue_name>_(read/offset/consume)_amount(MAX
 *                          (<QUEUE_NAME>_MAX_FAMILY_
 *  Thus, the read amount for the queue, qsamp, would b
 *  following array:
```

Figure 3. Autocoded Range Partition MPID 'C' Source

Observation	Hand Coded	Autocoded	Comment
Validation	Run time testing	MPID unit testing GrTT	GrTT testing performed by MCCI, MPID unit tests by LM/ATL
Number of Processors	7	8	8 required for memory only, processing load can be met with 7
Processing Time	<7sec/sec	6.662 sec/sec	
Development Time	6 man months	<3 man weeks	10x improvement
Integration and Test Time	10 man weeks	2 man weeks	5x improvement

Table 1. Comparison of Handcoded vs Autocoded Software Design

exercised using the GrTT test utility on input data supplied by the LM/ATL benchmark development team. MPIDs were tested using the unit tester and validated against the same test vectors. Figure 4 shows a comparison of GrTT behavior model testing and MPID source testing using the common input test vector. An application image was created using a prototype application generator. Input and output procedures were written to interface the software allocation to the hardware implementation of the input/output processing. These are user written procedures; however, input/output service routines provided with the tools were utilized that perform graph functions; e.g., enqueueing data. Two design iterations were completed during the evaluation period to optimize the autocoded implementation's performance on the available test hardware. The autocoded applications executed correctly without the need for a run-time test and fix effort.

3. Summary of Evaluation and Test Results

Table 1 summarizes the results the evaluation and compares them with the hand coded optimized implementation. MCCI's autocoded implementation of the benchmark required an additional processor for its memory. Features not included in the alpha version of the toolset will make memory usage comparable to the hand coded version when incorporated. GrTT testing was accomplished by MCCI independent of LM/ATL's evaluation. Results are compared with the validation testing accomplished by LM/ATL to illustrate its future integration.

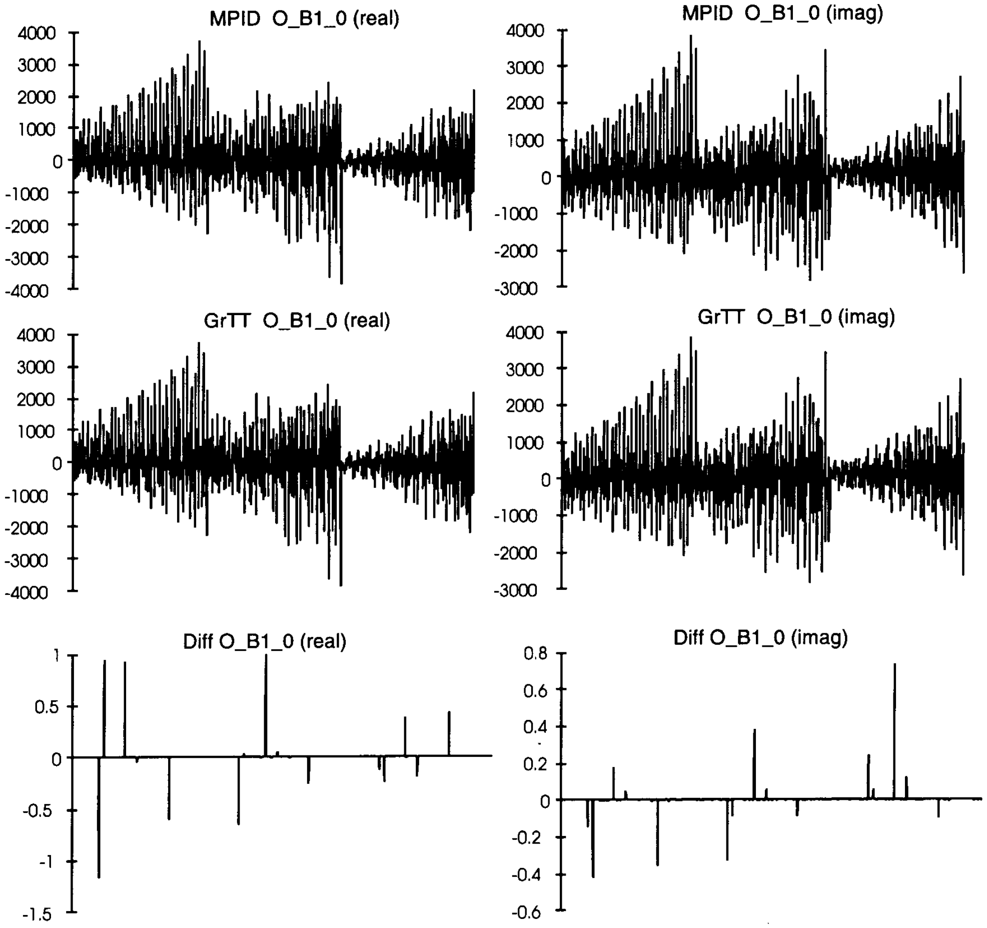


Figure 4. Comparison of Output Vectors from MPID Tester and GrTT Behavior Model

4. Progress Towards 4x Productivity Improvement

LM/ATL's evaluation of the alpha version of the autocoding tools demonstrated that MCCI's autocoding tools will generate application code with performance comparable to optimized hand coded implementations with an order of magnitude improvement in development productivity. This will meet the productivity enhancement goals required in the software generation elements in LM/ATL's productivity improvement model. Further improvements are anticipated as our tools mature. A reevaluation of the beta version of the autocoding tools is planned for July '96. Further improvement in run-time performance and productivity should be observed. We believe this additional improvement beyond the requirements will have beneficial synergistic effect with other elements of the codesign process.

Christopher B. Robbins
Management Communications
and Control, Inc.
2000 North 14th Street, Suite 220
Arlington, VA 22201
crobbins@mcci-arl-va.com

EaSE Trades Technical Review

Gary W. Panzer

Abstract

A methodology for the rapid, systematic, through and efficient exploration of very large digital processing design trade spaces has been developed. The resulting process is implemented as an Early System Evaluation and Trades (EaSE Trades) Design Advisor Tool. The Navy's Processing Graph Methodology (PGM) and JRS Integrated Design Automation System (IDAS) combined with the use of Design of Experiment (DOE) techniques for the experimental exploration of candidate design alternatives have achieved breakthrough improvements in the way trade studies are accomplished.

1. Objectives

The objective of RASSP is to improve the process by which embedded digital signal processors (DSP) are designed, manufactured, upgraded, and supported. This initiative has top-level goals of developing an advanced, systematic design capability to achieve

- 4x or better design and redesign speed improvement,
- 4x or better improvement in life cycle cost.

The RASSP design system relies heavily on virtual prototyping, that is, extensive simulation at increasing levels of fidelity. DSP system and architecture simulation tools, such as the JRS NETSYN are being developed to meet the virtual prototyping goals. These tools offer powerful methods for examining system alternatives early and often in the design process. They can be applied on problems covering a wide range of architectures, applications, and rates.

The capability of these new tools opens doors to exploration of vast alternate trade spaces. Even with the improvement in speed offered by these tools, thorough trade studies using simulation tools often cannot examine all potentially interesting trade conditions. Because of cost and time constraints, we are often forced to narrow the number of cases using engineering judgment or other subjective criteria. Important alternatives may be excluded before the trade analysis even starts. The solution lies in the use of a systematic methodology to search the trade space of alternatives.

In many commercial industries, the challenge of intelligently sampling a large trade space is accomplished using the field of statistics commonly referred to as Design of Experiments (DOE). By definition, DOE is the planned, structured, and organized observation of input (independent) variables (hereafter referred to as factors), and their effect on the output (dependent) variables (hereafter referred to as responses). In many industries and applications, DOE analysis has resulted in a considerable reduction in data collection necessary for exploration of large candidate trade spaces.

The Early System Evaluation and Trades (EaSE Trades) Program is developing a methodology for the rapid, systematic, thorough, and efficient exploration of large digital signal processing trade spaces. The process uses DOE as the basis for a Design Advisor that aids the engineer in the selection of simulation cases that will help to find the best trade choice quickly and efficiently.

2. Technical Approach

The EaSE Trades method is shown in Figure 1. It outlines the step-by-step process for rapidly searching the large, multidimensional design trade space of the DSP system. Sampling is done with confidence that all the potentially applicable trade conditions will be considered. The following paragraphs detail this process.

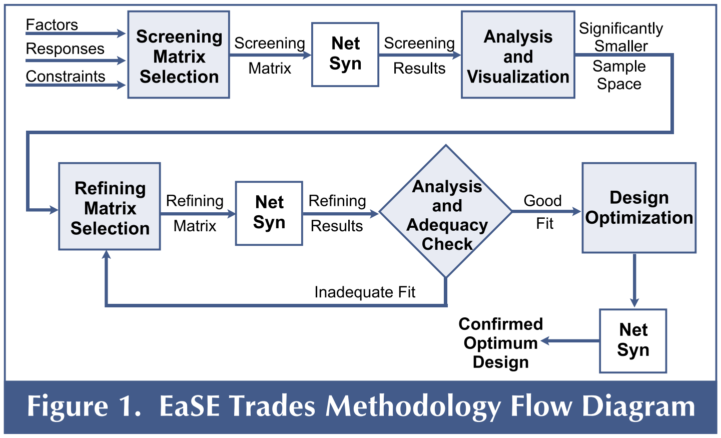


Figure 1. EaSE Trades Methodology Flow Diagram

Screening Matrix Selection - In most physical situations, some variables will be much more important and effective than others at meeting the customer's needs. DOE screening studies take advantage of this to quickly, and systematically, reduce the number of simulation cases that need to be examined. The user specifies the factors to be traded, the responses of interest, and any constraints on the problem. The EaSE Trades method uses sampling techniques designed for screening to intelligently choose simulation cases from the large trade space as represented by the factors being traded. The sampling plan recommends a matrix of test conditions to be simulated on NETSYN. The simulation results quantify the response values of interest.

Analysis and Visualization Refining Matrix Selection - The smaller trade space is then examined in a more refined experiment. DOE sampling techniques which are designed to produce a second order polynomial description of the response surface are used to create a matrix of test conditions to be simulated on NETSYN. A second order regression equation for each response of interest is derived from the refined experiment.

Analysis and Adequacy Check - The "goodness of fit" of the second order equation is tested statistically to assess the adequacy of the second order equation to describe the observed responses. If required, additional samples can be simulated to facilitate the use a higher order polynomial to describe the relations between factors and responses.

Design Optimization - Assuming a good fit, the second order polynomial equations are used to trade-off alternative solutions. The equations are solved for the solution that best meets the customer's needs. This optimum design is then simulated on NETSYN to confirm the predictions of the second order model.

The final product will be a confirmed optimum design for the DSP system.

3. Technical Results

Processing an experiment through the EaSE Trades Design Advisor produces results depicted in Table 1. This partial table lists the coefficients on an equation that predicts system performance. Coefficients of lesser importance have been removed from the table.

Table 1. Representative Equation Coefficients Factor Coefficient

Factor	Coefficient
Average (B_0)	41590
Processor P11 (B_1)	-10881
Processor P31 (B_3)	-3448
Processor P21* Processor P31 (B_{23})	3362
Processor P11* Processor P31 (B_{13})	3291
Processor P21 (B_{21})	-3286

The processing system's response equation is in the form:

$$Y = B_0 + B_1X_1 + B_{11}X_1^2 + B_2X_2 + B_{22}X_2^2 + B_3X_3 + B_{33}X_3^2 + B_{12}X_1X_2 + B_{13}X_1X_3 + B_{23}X_2X_3 + B_{123}X_1X_2X_3$$

Examining the above table, the negative coefficient values for the linear terms indicate a decreasing effect on the response time when the number of processors is increased. This is in agreement with our intuitive expectation. Processors P11 and P31 also appear as the most important in the response time. This is also in agreement with our expectations since they are the fastest processors as indicated by their specifications. Further analysis determined that the data behaved inverse square linearly, i.e., the reciprocal of the square of the response time gives the best fit. This confirms our original understanding of the system's 'inverse exponential' behavior.

4. Deliverables

The major deliverables of the program are

Methodology Definition Document - This material covers the technical aspects of the system operation for this program. It defines the overall strategy for conducting the research, and the approach to controlling the experiments and evaluating the results.

Integration and Test Plan - This documents the specific plan, including schedule, for mechanizing the tool interfaces, debugging and testing for proper operation prior to conducting methodology evaluations in the validation phase of the program.

Methodology Evaluation Plan - This is a specific plan how evaluations are to be conducted. It is intended to provide for a cost-controlled and disciplined research approach.

Final Report - This is a program and technical report that summarizes the work accomplished. Other deliverables will be included by reference or as appendices as specified by the contract.

Design Advisor Prototype - The Design Advisor software and documentation that is in-place at the conclusion of the research will be made available.

Gary W. Panzer
Hughes Radar and Communication Systems
P.O. Box 92426, RE/R01/A528
Los Angeles, CA 90009-2426
panzer@bala.hac.com

System-Level Design Methodology for Embedded Signal Processors

The Ptolemy Team[1]

1. Objectives

This project focuses on design methodology for complex real-time systems in which a variety of design methodologies and implementation technologies must be combined. Design methodologies are encapsulated in models of computation, whereas implementation technologies are implemented as synthesis tools. Applications that use more than one model of computation and/or more than one synthesis tool are *heterogeneous*. Hardware/software codesign is one example of heterogeneous design.

2. Technical Approach

The key idea in the Ptolemy project is to mix models of computation, implementation languages, and design styles, rather than trying to develop one all-encompassing technique. The rationale is that specialized design techniques are (1) more useful to the system-level designer and (2) more amenable to high-quality high-level synthesis of hardware and software.

3. Technical Contributions

We support heterogeneity by defining formal models of computation such as dataflow and finite-state machines at arbitrary levels of granularity, hierarchical mechanisms to compose models into complex systems, and algorithms to synthesize systems in hardware and software. We address

Algorithm Specification: graphical/algebraic specification [1]; algorithm restructuring [1]

System Specification: scalable systems; syntax management; integrated documentation; design methodology management [1]

Modeling: multidimensional dataflow [2]; finite state machines (FSM); synchronous/reactive (SR) controllers; process networks; formal mathematical analysis of dataflow models

Scheduling of Dataflow Graphs: joint minimization of program and data memory for embedded DSP processors [3]; optimization of synchronized communication between processors [3]; hierarchical scheduling [4]; incremental compilation [4]; dynamic scheduling

Simulation: compiled simulation [4]; mixing models of control (FSM, SR), dataflow, and discrete-event subsystems [5]; mixed-signal analog/digital simulation

Synthesis: converting (untimed) dataflow graphs into (timed) clocked circuits; fast partitioning algorithms for hardware/software codesign [6]

We test our ideas in the Ptolemy software environment. Ptolemy provides a framework for mixing tools with fundamentally different semantics and a laboratory for experimenting with these mixtures. Many different kinds of tools cooperate in the specification, simulation, and synthesis of systems. We mix our own custom tools with many existing ones to create a system-level design tool:

Block diagram specification: octtools design database; vem schematic editor; Ptolemy higher-order functions for scalable systems

Parameter specification: Ptolemy expression evaluator; Tcl; MATLAB; Mathematica; Tycho

Simulation: Ptolemy dataflow, discrete-event, and finite state machine domains; Ptolemy kernel-, MATLAB; Esterel; Tcl/Tk; Synopsys VHDL System Simulator; Model Technology VSIM VHDL Simulator; xgraph; xv; soundtool; Utah Raster Toolkit; GNU threads

Software synthesis: Ptolemy C, C++, and DSP assembly code generators; GNU and cfront C and C++ compilers; make; Motorola 56000 assemblers and simulators

Hardware synthesis: Ptolemy VHDL generation synthesized by Synopsys Design Analyzer.

4. Deliverables

During the RASSP contract, we published ninety papers and two books, released five versions of the Ptolemy software, initiated a news group, *comp.soft-sys-Ptolemy*, and created a Web site *http://Ptolemy.eecs.berkeley.edu/* for rapid dissemination of papers, software, documentation, and tutorials. Several other RASSP participants, such as Sanders, MIT/BU, and BDTI, have leveraged the Ptolemy software environment for rapid prototyping. Cadence has commercialized our ideas on dataflow modeling and scheduling in their SPW tool and our ideas on heterogeneous simulation in their CONVERGENCE architecture to allow the SPW and Bones simulators to cooperate.

The Ptolemy Project is broader than the RASSP contract. The Ptolemy Web site has a tutorial on the TI C30 DSP processor, a C code documentation extractor, and Signal Processing Packages for Mathematica, which Wolfram Research Inc. commercialized as the *Signals and Systems Pack.*

[1] Over the course of the RASSP contract the Ptolemy team has been directed by Prof. Edward A. Lee and co-directed by Prof. David Messerschmitt. Ptolemy software development was managed first by Alan Kamas and later by Christopher X. Hylands. Dr. Joseph T. Buck is the primary designer of the Ptolemy software architecture. The technical staff has included Dr. Brian L. Evans, Dr. Alain Girault, Dr. Takashi Miyazaki, Dr. H. John Reekie, Dr. Juergen Tcich, and several industrial scholars. The Ptolemy team also consists of Dr. Shuvra S. Bhattacharyya, Wan-Teh Chang, William Chen, Kang Ngee Chia, Stephen A. Edwards, Ron Galicia, Michael Goodwin, Steve X. Gu, Dr. Soonhoi Ha, Sangjin Hong, Farbad Jalilvand, Asawaree Kalavade, Karim P. Khiar, Joel King, Allen Lao, Bilung Lee, William Li, Xiao Mei, Praveen K. Murthy, Dr. Thomas M. Parks, Jose' Luis Pino, Farhana Shiekh, S. Sriram, Matthew Tavis, Warren W. Tsai, William Tsu, Patrick J. Warner, and Michael C. Williamson.

5. Selected References

[1] B.L. Evans, S.X. Gu, A. Kalavade, and E.A. Lee, “Symbolic Computation in System Simulation and Design,” Invited Paper, *Proc. of SPIE Int. Sym. on Advanced Signal Processing Algorithms, Architectures, and Implementations*, July 9-16, 1995, San Diego, CA, pp. 396-407.

[2] P.K. Murthy and E.A. Lee, “An Extension of Multidimensional Synchronous Dataflow to Handle Arbitrary Sampling Lattices,” *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, Atlanta, GA, May 7-10, 1996, vol. 6, pp. 3306-3309.

[3] Shuvra S. Bhattacharyya, Praveen K. Murthy, and Edward A. Lee, *Software Synthesis from Dataflow Graphs*, Kluwer Academic Press, ISBN 0-7923-9722-3, April, 1996.

[4] J.L. Pino, M.C. Williamson, and E.A. Lee, “Interface Synthesis in Heterogeneous System-Level DSP Design Tools” *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, Atlanta, GA, May 7-10, 1996, vol. 2, pp. 1268-1271.

[5] W.T. Chang, S.H. Ha, and E.A. Lee, “Heterogeneous Simulation - Mixing Discrete-Event Models with Dataflow,” Invited Paper, *Journal on VLSI Signal Proc.*, RASSP special issue, to appear.

[6] A. Kalavade and E.A. Lee, “The Extended Partitioning Problem: Hardware/Software Mapping and Implementation-Bin Selection,” *Proc. IEEE Int. Work. on Rapid Sys. Prototyping*, June 1995, pp. 12-18.

The Ptolemy Team
Department of Electrical Engineering
and Computer Sciences
University of California, Berkeley, CA 94720-1772
http://ptolemy.eecs.berkeley.edu/

Numeric and Symbolic Algorithms for Signal Processing

Alan Oppenheim

Abstract

The overall goals of the RASSP Technology Base program at the Massachusetts Institute of Technology involve the areas of algorithms (numeric and symbolic) design for signal processing applications. Specifically, the task focuses on identifying and developing approximate classes of emerging and classical algorithms which will exercise and challenge RASSP tools and process.

1. Objectives

The following were the main technical objectives of our RASSP program at MIT.

1. Develop and analyze algorithms for approximate signal processing and incremental refinement.
2. Develop and analyze algorithms for low-power signal processing.
3. Develop and implement signal processing systems using both numeric and symbolic processing representations.
4. Develop and analyze new methods for algorithm-based fault tolerance.
5. Investigate novel algorithms based on emerging mathematical paradigms.

We will now describe each of these objectives and their associated activities briefly.

2. Activities and Accomplishments

2.1 Approximate Algorithms

In the area of approximate algorithms, a number of incremental refinement (IR) algorithms have been proposed for various DSP transforms. Specifically, an IR approach was developed for FFT-based Maximum Likelihood (ML) detection. We have also recently extended previous work on an IR approach to approximation of the Discrete Fourier Transform/Short Time Fourier Transform (STFT) by analyzing the probability of achieving a desired tradeoff between cost and quality for a given class of signals. We have also extended these algorithms to support mixed-radix signal representations, allowing a greater computational efficiency to be achieved in the IR context.

Our current activities in this area are directed to obtaining specific performance results for IR FFT-based ML detector on real sinusoids in noise, and considering techniques for reducing memory and hardware requirements for the IR two-dimensional Inverse Discrete Cosine Transform (IDCT) [3-5].

2.2 Low Power Signal Processing

In the area of low power design, we are investigating algorithmic and architectural approaches to reformulating algorithms to minimize power consumption. We have examined the tradeoffs between accuracy and power consumption and demonstrated low power implementations in FIR and IIR filters using adaptive approximate processing concepts. We have also investigated the links between "activity" and optimum ordering of filtering operations.

This work has resulted in new techniques that dynamically adjust filter order based on signal statistics, and demonstrate low power filter implementations. A Matlab-based simulation was also developed to evaluate transition activity of various ordering schemes.

Our future activities in this area include methods to quantify power reduction in IIR filter structures, and apply low power techniques to image coding. Extensions to adaptive wordlengths are also to be investigated. Tradeoffs between memory-based (lookup) and arithmetic-based signal processing will be evaluated from these viewpoints.

2.3 Numeric/Symbolic Signal Processing

In this area, we and our subcontractors at Boston University led by Professor Nawab, have investigated the design and implementation for a variety of DSP algorithms. The application of knowledge-based control to signal processing systems has been studied, resulting in a Integrated Processing and Understanding of Signals Architecture (IPUS) within existing design environments [6].

A number of new developments include upgrading our IPUS C++ platform to support full IPUS functionality. We have integrated this into the Ptolmey design environment from UC Berkeley to obtain an IPUS domain. A number of testbeds for clutter analysis have been implemented within Ptolemy, using the IPUS domain.

Our future activities in this area are to release the IPUS domain for Ptolemy in version v.0.6 in Summer 1996. We will also develop a support for heterogeneous simulation with the IPUS domain using Ptolemy's wormhole feature.

2.4 Algorithm-based Fault Tolerance

Researchers at the University of Illinois (Abraham et al, 1984) proposed algorithm-based fault tolerance (ABFT) to protect algorithms against faults using the following steps:

Step 1: Encode operands to introduce redundancy.

Step 2: Modify algorithms for encoded data.

Step 3: Check for and correct codes.

Step 4: Decode.

Our approach is based on the following steps:

- Recognize algebraic structure in algorithms (e.g., groups, rings, vector spaces, semigroups and dynamical computational processes).
- Encode by well-behaved algebraic mapping to larger algebraic set to introduce redundancy (Step 1 above).
- Alegebraic structure then points to necessary modification of the algorithms (Step 2), error checking and correcting (Step 3), decoding (Step 4).
- Extract hardware implementations.

A number of accomplishments include algebraic tools and methods for incorporating error detecting and correcting capabilities into nonlinear signal processing algorithms. We have also characterized all parity-type codes for fault-tolerant computations for certain algebraic structures, and included mechanisms to support ABFT in hardware, [8].

2.5 Emerging Mathematics for Signal Processing

In this area we have studied non-linear signal processing algorithms for the application of chaotic systems, soliton systems, and stochastic resonance systems and their implication in the development of DSP systems. This has also been directed at implementation aspects of next-generation wireless communications, a "spread-signature CDMA" system that combats strong multipath fading [1,2,5].

Our current work at the MIT RASSP group in this area includes developing fractal models for data traffic and the development of protocols for efficient traffic management. We will also continue to examine real-time DSP-based implementation of spread-signature CDMA systems for wireless applications, and explore the use of stochastic resonance systems in noise suppression [7].

3. Summary

Significant accomplishments in the areas listed above have been documented in technical reports, papers, and software releases as outlined in this article. Refer to the RASSP server for further information on the MIT Technology Base program.

4. Selected Publications

[1] K. Cuomo, A. Oppenheim, S. Strogatz, "Robustness and Signal Recovery in a Synchronized Chaotic System", International Journal of Bifurcation and Chaos, Vol. 4, No. 6, pp 1629-1638, 1993.

[2] E. Weinstein, A. Oppenheim, M. Feder “Iterative and Sequential Algorithms for Multisensor Signal Enhancement,” IEEE Trans. on Signal Processing, Vol 42, No. 4, April 1994.

[3] J. Winograd, S. H. Nawab, “Incremental Refinement of DFT and STFT Approximations,” IEEE Signal Processing Letters, February 1995.

[4] S. Nawab and E Dorken, “A Framework for Quality vs Efficiency Tradeoffs in STFT Analysis,” IEEE Trans. on Signal Processing, April 1995.

[5] G. Wornell, “Spread-Signature CDMA: Efficient Multiuser Communication in Presence of Fading,” IEEE Trans. on Information Theory, September 1995.

[6] J. Winograd, S. Nawab, "A C++ Software Environment for the Development of Embedded Signal Processing," IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, ICASSP-95, Detroit, 1995.

[7] J. Winograd, S. Nawab, A. Oppenheim, "FFT-based Incremental Refinement of Suboptimal Detection," IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, ICASSP-96, Atlanta, 1996.

[8] C.N. Hadjicostis and G.C. Verghese, "Fault-Tolerant Computation in Semigroups and Semirings," submitted for journal review, 1996.

A. V. Oppenheim
DSP Group, RLE
MIT
Cambridge, MA 02139
avo@allegro.mit.edu

COMET Project: Hardware/Software Cosynthesis for DSP Systems

Ranga Vemuri

1. Project Objectives

The goal of the COMET project is to develop languages, techniques and tools for hardware, software cosynthesis of board- and MCM-level DSP (Digital Signal Processing) systems from very high level requirements specifications.

COMET target architectures can be either self-contained embedded processors in a primarily non-computing environment or special purpose co-processors that perform compute intense tasks delegated by a master computer system. In both cases, the DSP architecture may contain custom hardware components, custom software components executing on an off-the-shelf DSP/GP processor or both. Hardware components may in turn be application specific integrated circuits (ASIC), multichip modules (MCMs), field programmable gate arrays (FPGA) or a combination of these, mounted along with off-the-shelf parts on a printed circuit board (PCB).

2. The COMET Design Environment

Typical top-down design of the target architectures begins with requirements specification and analysis and continues to the design synthesis of both hardware and software components. COMET architecture is centered around (1) VSPEC, a declarative interface requirements specification language for VHDL entities, (2) hardware/software partitioning techniques for embedded DSP systems, and (3) codesign performance analysis methods and tools. COMET design environment is shown in Figure 1.

COMET design tools can be used in a variety of design processes or as stand-alone tools. The flow in Figure 1 suggests a top-down cosynthesis approach from high-level requirements specifications.

3. COMET Technology

3.1 Requirements Specification

VSPEC is the requirements specification language of COMET. VSPEC is a declarative annotation language for VHDL entities. Through VSPEC, designers specify requirements the system design should meet and constraints on its implementation. A VSPEC specification consists of a collection of logical statements and declarations that annotate a VHDL **entity** construct.

VSPEC adds seven clauses to a VHDL entity that allow a specifier to define "what" the entity must do without defining "how" it will be done. The **requires, ensures** and **sensitive to** clauses are used to specify the functional requirements of the device. Non-functional constraints are described in the **constrained by** and **modifies** clauses. The internal state of the component is declared in the **state** clause and the **includes** clause is used to make predefined types and operators visible in a VSPEC component [1].

VSPEC specifications can be used to evaluate component reusability. Specifications are used to compare the behavior of existing components to the requirements of a new system. Effective component reuse requires a tool for finding potential reuse candidates from within a component library. A component classification and retrieval tool, named REBOUND, is developed to provide efficient retrieval of VSPEC components [2].

3.2 Hardware/Software Partitioning

The goal of system partitioning is to generate a first level hardware-software architecture of the system by partitioning the system specification into specifications of hardware components and software components. The hardware components will be further processed by hardware synthesis tools. The software components will be bound to execute on a selected DSP or general purpose processor configuration. The hardware and software components will be connected to constitute a VSPEC - VHDL architectural description of the system. The functional requirements and constraints stated in the VSPEC specification drive the derivation of the specific hardware-software mix.

Initially, the VSPEC system specification is refined based on queries into the design library. As a result of the queries, components are selected based on their ability to satisfy the system function and constraint attributes. In case the existing components do not meet the requirements, a design that partially satisfies the requirements may be generated. Alternatively, the designer may be queried for additional components. Scoreboard algorithm for hardware software partitioning and binding is based on the iterative improvement and allows inclusion new constraints as they arise.

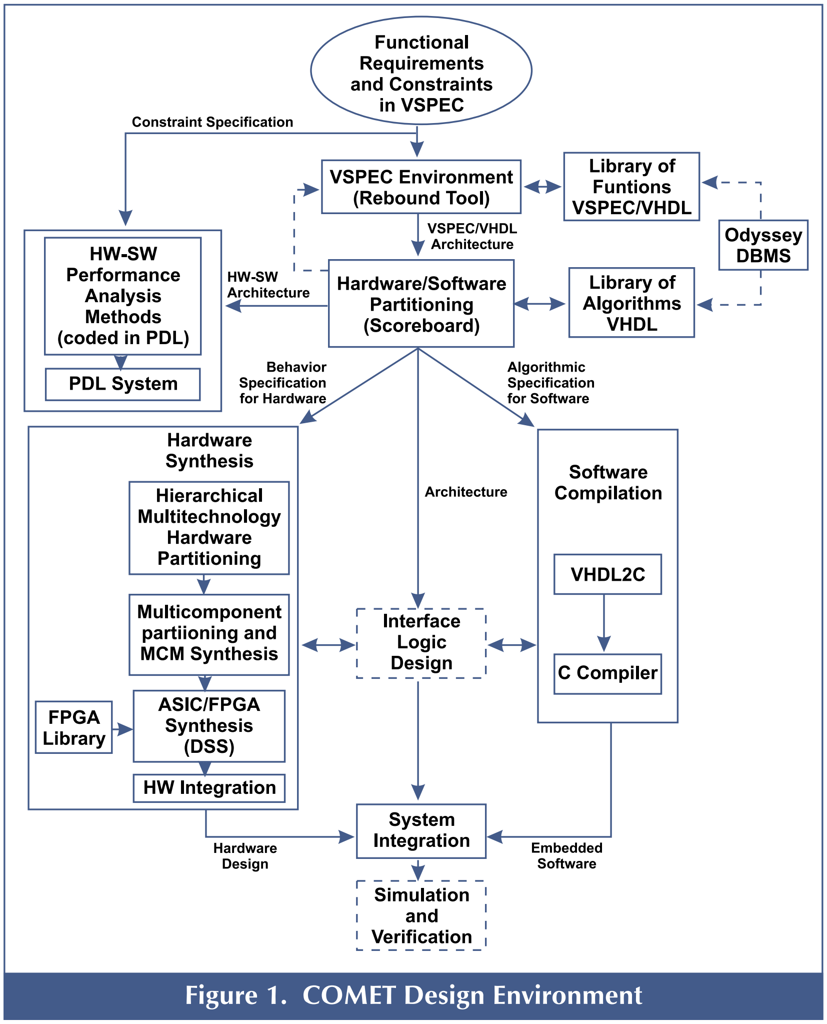

Functional Requirements and Constraints in VSPEC
Constraint Specification
VSPEC Environment (Rebound Tool)
Library of Funtions VSPEC/VHDL
Odyssey DBMS
VSPEC/VHDL Architecture
HW-SW Performance Analysis Methods (coded in PDL)
PDL System
HW-SW Architecture
Hardware/Software Partitioning (Scoreboard)
Library of Algorithms VHDL
Behavior Specification for Hardware
Algorithmic Specification for Software
Hardware Synthesis
Hierarchical Multitechnology Hardware Partitioning
Multicomponent partiioning and MCM Synthesis
FPGA Library
ASIC/FPGA Synthesis (DSS)
HW Integration
Architecture
Interface Logic Design
Software Compilation
VHDL2C
C Compiler
Hardware Design
System Integration
Embedded Software
Simulation and Verification

Figure 1. COMET Design Environment

3.3 Codesign Performance Analysis

Accurate performance estimation is critical to the success of a design synthesis system. The COMET performance estimator is used to evaluate the performance, in terms of area, speed, throughput rate, and power dissipation, of the library components as well as the performance of a contemplated hardware-software architecture of a system. The estimator can be used interactively or through the partitioning engine to filter inferior architectures and to select a constraint-satisfying hardware-software binding for a given specification. Various hardware-software alternatives can be selected for each component in the architecture and for each selected configuration performance envelopes can be generated.

COMET performance estimator is based on the PDL (Performance Description Language) system for performance modeling and analysis. Tradeoff analysis is a central aspect in the codesign process. PDL system supports performance analysis and tradeoff visualization for rapid prototyping of codesigns [3, 4].

3.4 Hardware Partitioning and Synthesis

COMET hardware synthesis system consists of a multicomponent partitioning engine and a set of synthesis tools for ASIC , FPGA and MCM synthesis. The multicomponent partitioning engine [5, 6] is a hierarchical partitioning and package binding tool that accepts behavioral specifications in VHDL, constraints on area, power consumption, pin counts, speed and cost and generates a hierarchical partition of the specification with each component in the partition bound to a package among a set of available packages. The partitioning engine uses a back-tracking algorithm for constraint- directed search. Power estimation is based on data gathered by dynamic profiling of the VHDL specification using typical stimuli.

The ASIC synthesis system DSS (Distributed Synthesis System) [7] accepts behavioral specifications in VHDL and constraints on clock period and area. It generates register level designs in VHDL. Register level designs contain two parts: a data path and a finite state controller. The data path is in the form of structural VHDL in which each component is instantiated from a predefined parameterized register level component library.

MCM synthesis environment MSS [8] is embedded in COMET to facilitate synthesis of multichip modules. Multichip designs can be generated in two ways: (1) Register level designs generated by DSS can be partitioned into multiple chip designs, or (2) an integrated behavior synthesis and partitioning step can be performed to obtain multichip designs directly. These multichip designs are then processed by package level place and route tools. We currently use Mentor Graphics MCM Station.

3.5 Software Compilation

The software synthesis tools in COMET translate DSP-based software behavioral specifications expressed in a subset of VHDL into efficient machine code. The overall approach to software synthesis is to translate behavioral descriptions expressed in VHDL into C and then use commercial C compilers to translate C into machine code to execute on the target processor. The currently supported processors are the Texas Instruments TMS430C51, Sun Microsystems SPARC, and Intel 80386. The compiled code can be statically analyzed for timing performance to ensure compliance with timing constraints expressed in the VSPEC specification.

The VHDL subset used as input for software synthesis is similar to that used for asic synthesis [7]. VHDL behavioral constructs are fully supported along with a limited subset of structural constructs. Explicit timing , such as VHDL **after** clauses or specific time in *wait* statements, is not supported.

The execution environment consists primarily of a small multitasking operating system kernel which will provide interprocess communication service, task management, and input/ output support. The task scheduler will create, maintain and monitor all tasks in the run-time space, while the interprocess communication protocol will support a simple message passing mechanism where a process writes its request and data in a message channel whenever it tries to communicate with others, and then optionally waits until a response is received.

4. Availability of COMET Technology and Tools

COMET tools are freely available to the RASSP community. Some of the COMET tools (DSS behavioral synthesis tool, MSS multicomponent synthesis tool, PDL performance modeling and analysis environment, VHDL to C compiler, VSPEC parser) are relatively mature. Other tools (Rebound, Scoreboard) are under development with release versions expected shortly. Triquest Design Automation Inc., is in the process of receiving a subcontract to quality enhance and distribute some of the COMET tools.

A WAVES Level 2 usage guide is available to aid in the preparation of test benches in the IEEE standard WAVES language [9]. COMET tools have been successfully used to synthesize various designs including reconfigurable coprocessors [10]. Further information on the COMET project is available through worldwide web page *http://www.ece.uc.edu/~ ddel/comet.html*.

References

[1] P. Baraona, J. Penix, and P. Alexander, “VSPEC: A Declarative Requirements Specification Language for VHDL,” Current Issues in Electronic Modeling, vol. 3, pp. 51-75, 1995.

[2] J. Penix, P. Baraona, and Perry Alexander, “Classification and Retrieval of Reusable Components Using Semantic Features,” Proc. 10th Knowledge-Based Software Engineering Conference,” pp. 131-138, 1995.

[3] R. Vemuri, R. Mandayam, and V. Meduri, “Performance Modeling Using PDL,” IEEE Computer, April 1996.

[4] J. Walrath, et al., “Performance Modeling and Tradeoff Analysis During Rapid Prototyping,” Proc. Application Specific Array Processors, 1996.

[5] N. Kumar, V. Srinivasan, and R. Vemuri, “Hierarchical Behavioral Partitioning for Multicomponent Synthesis,” Proc. European Design Automation Conference, 1996.

[6] M. Vootukuri, R. Vemuri and N. Kumar, “Partitioning of Register Level Designs for Multi-FPGA Synthesis,” Proc. VIUF Conference, Spring 1996.

[7] J. Roy, R. Dutta, N. Kumar, and R. Vemuri, “DSS: A Distributed Synthesis System for VHDL Specifications,” IEEE Design and Test of Computers, pp. 18-32, June 1992.

[8] R. Vemuri et al., “An Integrated Multicomponent Synthesis System for MCMs,” IEEE Computer, pp. 62-74, April 1993.

Ranga Vemuri
University of Cincinnati
ECECS Department, M.L. 30
Cincinnati, Ohio 45221-0030
ranga.vemuri@uc.edu

ADEPT: A Unified Environment for System Design

Moshe Meyassed, Bob McGraw, Robert Klenke, James Aylor, and Barry Johnson

Abstract

This paper presents an unified end-to-end design environment that supports the design of digital systems from initial concept down to the final implementation. A tool called ADEPT (Advanced Design Environment Prototyping Tool) has been developed to implement this environment. ADEPT supports both system level performance and dependability analysis in a common design environment using a collection of predefined library elements, called ADEPT modules. These modules have a VHDL description as well as an underlying colored Petri Net representation associated with them. As a result, both simulation-based and mathematical approaches for analysis can be employed. A novel aspect of ADEPT is the ability to simulate both uninterpreted (performance) and interpreted (behavioral) models in a common simulation environment using a technique called hybrid modeling. Hybrid modeling allows the stepwise refinement of system level models into implementation level models. The ADEPT design tool is available via anonymous ftp.

1. Introduction

An end-to-end unified design environment based upon the use of VHDL is being developed by the Center for Semicustom Integrated Systems (CSIS) at the University of Virginia. This environment also has a mathematical basis in Petri Nets thus providing the capability for analysis through simulation or analytical approaches. A tool set called ADEPT (Advanced Design Environment Prototyping Tool) has been developed to implement this environment. ADEPT supports the integrated performance and dependability analysis of system level models. ADEPT is also being extended to support operational specification modeling and hardware/software codesign. Additionally, ADEPT has the capability to simulate both uninterpreted and interpreted models in a common simulation environment using a technique called *hybrid modeling*. Hybrid modeling allows the stepwise refinement of system level models into implementation level models.

This unified design environment provides several advantages over previous design environments. First, a common modeling language that spans numerous design phases is much easier to use, encouraging more design analysis and consequently better designs. A common modeling language enables the possibility of a common simulation environment. The common simulation environment decreases the need for translators and multiple design environments which reduces inconsistencies and the probability of errors in translation and greatly speeds the design process. Second, the existence of a mathematical foundation within the unified environment provides the capability for complex system analysis using analytical approaches.

The major components of the ADEPT environment are a library of predefined elements (called ADEPT modules) from which system level performance and reliability models can be constructed, and a set of tools for automating the construction, simulation, and analysis of these system level models.

2. The ADEPT Modules

In the ADEPT environment, a system model is constructed by interconnecting a collection of ADEPT modules. The modules model the information flow, both data and control, through a system. Each ADEPT module has an associated VHDL behavioral description and a corresponding Colored Petri Net (CPN) representation. These CPN descriptions can be used to reduce the system model size and consequently reduce the overall simulation time of the model. In addition, the CPNs can be transformed into Markov models from which dependability measures (reliability, availability, safety, etc.) can be obtained using well known analytical techniques.

The ADEPT modules communicate by exchanging tokens, which represent the presence of information, using a uniform, well defined handshaking protocol. Higher level modules can be constructed from the basic set of ADEPT modules. In addition, custom modules can be incorporated into a system model as long as the handshaking protocol is adhered to.

A token is implemented as a record in VHDL. In the token, the first element in the record is the STATUS field of the token and the second element is an array of user defined COLOR fields. The STATUS field is used to implement the token passing mechanism, that is, the "handshaking" between the ADEPT modules. The COLOR fields contain up to 18 integer tags that can hold user-specified information. Modules are provided which can manipulate the information in the COLOR fields.

The basic set of ADEPT modules is divided into six categories: control modules, color modules, delay modules, fault modules, miscellaneous parts modules, and hybrid modules. The control modules, such as the *Switch*, and *Queue* modules, store and control the flow of tokens. The color modules enable the manipulation of the token color fields and the delay modules model the temporal aspects of a system. The fault modules are used to model the presence and detection of faults and errors in a system model. The miscellaneous parts category contains modules that are used for data collection within the ADEPT system. The hybrid category contains modules which aid in the construction of hybrid models, as explained below.

Because ADEPT uses VHDL as its simulation language, it provides the designer with the unique ability to simulate uninterpreted (performance) models with interpreted (behavioral) models in a common environment, termed *hybrid modeling*. Hybrid modeling allows the modeler to cosimulate elements which are modeled at different levels of design detail within the same model. The hybrid modeling interfaces within ADEPT are mainly concerned with handling the performance (uninterpreted) modeling to functional/behavioral (interpreted) modeling transition. The hybrid modeling interface must handle the actual conversion of the tokens from the uninterpreted model into the values (integers, reals, bits, etc.) required by the interpreted model and the “filling in” of the information in the interpreted model that may not be present in the uninterpreted model.

The hybrid category of ADEPT modules contains the elements necessary to construct the hybrid interfaces. Modules are included to construct hybrid interfaces between uninterpreted models and interpreted models with both combinational and sequential elements.

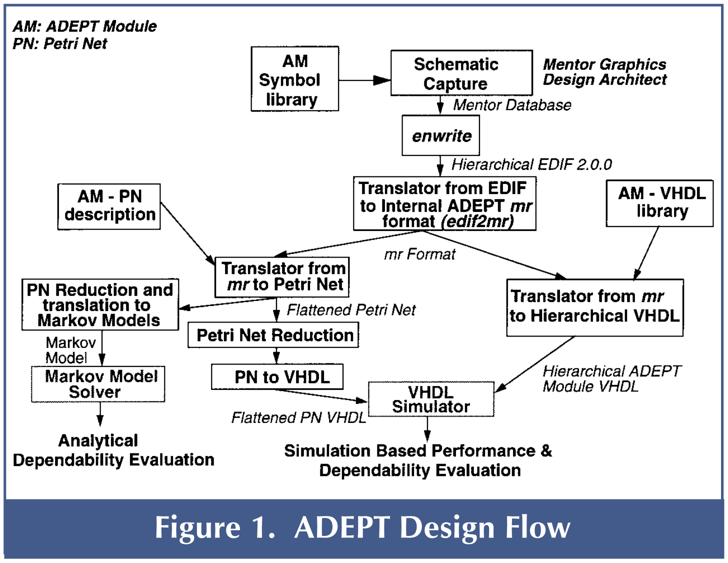

AM: ADEPT Module
PN: Petri Net
AM Symbol library
Schematic Capture
Mentor Graphics Design Architect
Mentor Database
enwrite
Hierarchical EDIF 2.0.0
AM - PN description
Translator from EDIF to Internal ADEPT mr format (edif2mr)
AM - VHDL library
mr Format
Translator from mr to Petri Net
PN Reduction and translation to Markov Models
Flattened Petri Net
Translator from mr to Hierarchical VHDL
Petri Net Reduction
Markov Model
Markov Model Solver
PN to VHDL
Hierarchical ADEPT Module VHDL
VHDL Simulator
Flattened PN VHDL
Analytical Dependability Evaluation
Simulation Based Performance & Dependability Evaluation

Figure 1. ADEPT Design Flow

3. The ADEPT Front-end Tools

The overall architecture and design flow of the ADEPT tool set is shown in Figure 1. The ADEPT system includes a graphical front-end that the designer can use to construct system level models using ADEPT modules. The graphical front end is currently based on the Mentor Graphics’ Design Architect schematic capture system. Once the schematic is constructed using Design Architect, it is exported to the ADEPT tool set as EDIF 2.0.0. The ADEPT tools then translate the EDIF into either a hierarchical VHDL netlist representation or a CPN representation. The user can simulate the hierarchical VHDL that is obtained by utilizing the behavioral VHDL descriptions of the ADEPT modules. Alternatively, the equivalent CPN can be used to perform analytical model reduction or dependability analysis functions.

Figure 2 is a sample DA screen showing an ADEPT model. The schematic shown is that of a Triple Modular Redundancy (TMR) system with three dedicated repair processes. Most of the elements in this top level schematic are hierarchical, with separate design “sheets” describing how each component is constructed from ADEPT modules. The various components are graphically interconnected with signals. Once constructed, a model such as the one shown in Figure 2 can be analyzed for performance and dependability measures using the ADEPT tools described below.

4. The ADEPT Analysis Tools

System level models in ADEPT can generate data that can be analyzed to determine system performance and dependability measures. In ADEPT, the miscellaneous parts category contains modules that collect data on the token flow during the model simulation. Several post processing tools are included in the ADEPT tool set that can generate performance metrics such as throughput, utilization, latency, and queue lengths from the information captured by the miscellaneous parts modules. These tools can display the performance information either as text, or as an animated graphical display.

In ADEPT, fault modules can be inserted into a system level model to measure the dependability attributes of a system. Dependability analysis can be undertaken in ADEPT using either simulation based or analytical approaches. In the simulation based approach, the ADEPT model, with the fault modules inserted, can be directly simulated for various mission times to gather statistics on dependability metrics. However, because of the typically low failure rates used in fault modeling, significant simulation time may be required to accurately measure the required dependability metrics. Alternatively, a version of the ADEPT fault modules has been created that allows the ADEPT model to be interfaced to the

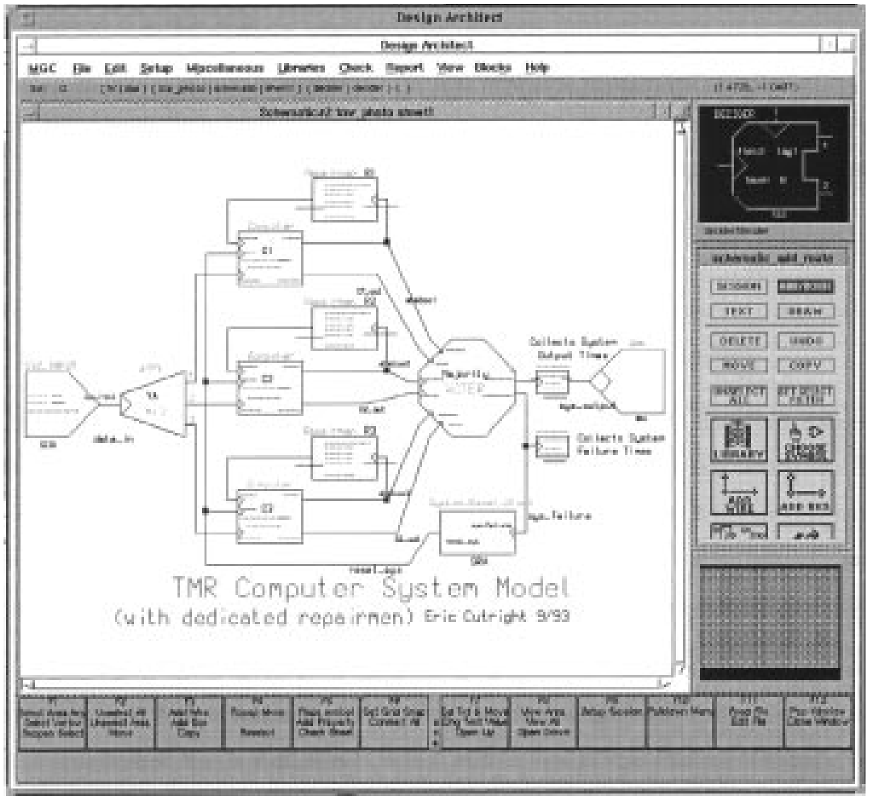

Design Architect
MGC File Edit Setup Miscellaneous Libraries Check Report View Blocks Help
Collects System Output Times
Collects System Failure Times
TMR Computer System Model
(with dedicated repairmen) Eric Cutright 9/93

Figure 2. Sample DA Screen

Reliability Estimation System Testbed (REST) developed by NASA. The ADEPT-REST interface allows the REST engine to manipulate the ADEPT model in such a way that it can create a Markov model of the system based on operational and failed system states and then analyze it to determine the dependability metrics of the system.

In addition to the simulation based approaches which use the VHDL model of the system, the CPN representation can also be used to perform analytical reliability analysis. The CPN description of the system model is reduced by removing all information in the model that does not affect reliability analysis. The resulting reduced CPN is then translated into a Markov model. The Markov model can then be solved using standard techniques to obtain dependability information.

Finally, a tool called *AnimateAdept* is included in the ADEPT tool set. *AnimateAdept* provides the ability to view the flow of tokens in the model using the Design Architect schematic. The status of tokens is represented by four different colors, and signals in the schematic change their color dynamically according to the temporal status of the tokens flowing through them. Additional information, such as user-defined color information, can also be presented dynamically on the *screen. AnimateAdept* provides a quick and easy way to debug models and understand the way in which phenomenon like bottlenecks and deadlocks are formed.

5. Application-Specific Libraries

The basic set of ADEPT modules can be used to model almost any type of system. However, the ADEPT modules include a fairly low level of functionality and thus large system models constructed from them may have some simulation performance or model complexity problems.

Fortunately, ADEPT allows the generation of application specific libraries of modules. This feature provides the ability to create libraries of modules which target modeling certain types of systems. By utilizing such a library, the user can save a significant amount of time constructing and simulating different models in the same application area.

Several such libraries have been created and included in the standard ADEPT environment. One such library includes elements for modeling digital communication networks. Five different communication networks, ATM, Ethernet, Myrinet, Raceway Interconnect, and the SCI, were used to guide the development of this library. This communication library contains four types of ADEPT modules: transmitters, receivers, routers, and bus routers. The transmitter modules are responsible for taking a particular message at their input and formatting and packetizing this message according to the communication protocol being modeled. The transmitter then sends the proper packets through the communications network. The receiver is responsible for the converse of the transmitter, it takes the packet data from the network and reconstructs the original message to be sent to its output. Separate transmitter and receiver elements exist for each protocol that is included in the library. The router is responsible for routing the packet data from one of its inputs to one or all of its outputs. The router can be parameterized for modeling all of the networks mentioned with the exception of the Ethernet in which case a bus router must be used. The bus router performs the same function as the router, however, only one message may be present on the bus router at a time. With the development of this communication library ADEPT users can now create models of communications networks quickly and easily.

A library of modules in ADEPT geared towards modeling cycle-based synchronous systems such as state machines and microprocessors is currently being developed. The cycle-based modeling environment employs techniques which enforce synchronization requirements at clocked memory elements. This library contains abstract models of the combinational devices (special purpose logic, ALUS, routing elements) and sequential devices (synchronous and asynchronous memories, register files) which are commonly used for processor development. Analysis techniques are also being developed to evaluate different architectures at an abstract level of design. Such techniques include tests for minimum cycle time, tests for determining the optimal amount of microconcurrency for a given instruction, throughput analysis, utilization analysis and functional analysis.

6. Obtaining ADEPT

The current version of ADEPT is available via anonymous ftp. For information on obtaining ADEPT, consult the ADEPT home page: *http://csis.ee.virginia.edu/~rassp/adept.html.* ADEPT currently runs on a Sun SPARC platform and requires Mentor Graphics' Design Architect as a front end. However, interfaces to other schematic capture tools, such as ORCAD and CADENCE, are under development.

7. Related Publications

Additional information on ADEPT related research can be found in the following papers:

[1] S. Kumar, R.H. Klenke, J.H. Aylor, B.W. Johnson, R.D.Williams, and R. Waxman, "ADEPT: A Unified Environment for End-to-End System Design," *Journal of Current Issues in Electronic Modeling.*

[2] R. Rao, A. Rahman, and B.W. Johnson, "Integrated Performance and Dependability Analysis Using the Advanced Design Environment Prototype Tool ADEPT," *Proceedings of the AIAA Computing in Aerospace Conference*, San Antonio, TX, 285-300, March, 1995.

[3] M. Meyassed, R. McGraw, J.H. Aylor, R.H. Klenke, and R.D. Williams, "A Framework for the Development of Hybrid Models," *Proceedings of the 2nd Annual RASSP Conference*, July, 1995.

[4] S. Kumar, J.H. Aylor, B.W. Johnson, and W.A. Wulf, "A Framework for Hardware/Software Codesign," *IEEE Computer*, vol. 26, 39-45, December, 1993.

[5] G. Swaminathan, J.H. Aylor, and B.W. Johnson, "Model Reduction Techniques Using Colored Petri Nets," *Proceedings of the 1993 TECHCON*, Atlanta, GA, 291-293, October, 1993.

Moshe Meyassed, Bob McGraw, Robert Klenke, James Aylor, and Barry Johnson
Center for Semicustom Integrated Systems
Department of Electrical Engineering
University of Virginia
mm8u, rmm2d, rhk2j, jha, bwj @Virginia.EDU

Performance Modeling Workbench - A VHDL-Based Hardware/Software Codesign Tool

Charles W. Buenzli, Jr.
and Jay Runkel

Abstract

Omniview, Inc., in conjunction with the Honeywell Technology Center, is developing a VHDL based Hardware/Software Codesign tool, code named Performance Modeling Workbench (PMW), as part of the RASSP Technology Base program. PMW allows a designer to rapidly create alternative hardware/software architectures and simulate them to validate system performance, find bottlenecks, and identify overdesigns. PMW facilitates the RASSP 4X goal by reducing performance model development time by an order of magnitude and by surfacing problems early in the design cycle.

1. Introduction

Incremental refinement and validation of each design phase through VHDL modeling is one of the cornerstones of the RASSP design methodology. VHDL performance models, because of their high level of abstraction, play a key role in this process during the early system design phases (see Figure 1). They also bridge the communication gap between system design and the detailed design of the hardware and software.

Performance models describe a system's time-related aspects: throughput, response time, latency, and utilization. They generally do not model the applications data or its transforms except for some high level control related behavior. As a result, performance models simulate much faster than behavioral models. In one example [1], a 24 processor system modeled in VHDL at the performance-level simulated the equivalent of 2,857,000 instructions per second, versus 5 instructions per second for a single I860 processor modeled at the Instruction Set Architecture (ISA)-level model.

The major impediments to the widespread adoption of VHDL performance modeling bus been the lack of commercially available performance models in VHDL and a user-friendly environment that allows the designer to model systems in familiar terms and isolates him from the underlying implementation details. The Honeywell Technology Center has partially solved the model availability problem with their development of a generic, parameterized library of VHDL-based performance models. How ever, most users found them too complex to understand and use [2-5]. It is Omniview's vision and goal for PMW to provide an integrated, user-friendly environment for VHDL-based performance modeling that will remove these impediments and efficiently analyze the volume of performance data from systems having hundreds or thousands of processors.

2. PMW Overview

The PMW enables designers to rapidly construct performance models by providing a library of reusable performance-modeling elements. This library is based upon the Honeywell Performance-Modeling Library and leverages off the performance-modeling techniques developed at the University of Virginia [6]. The PMW library, which is implemented in VHDL, contains detailed performance models of processors, memories, communication devices, input and output devices, operating systems, and application software (Figure 2). These models define the speed at which processors execute various classes of instructions, characterize the sequence of instructions representing software applications, and the speed at which memories, input and output devices, and communication elements process data. In addition, these models may be simulated with fully-functional models supporting an all-VHDL, design process where the functional blocks of a performance model can be incrementally refined to functional models as necessary.

The PMW provides a suite of design editors for configuring performance models that combine elements from the PMW performance-modeling library (Figure 3). A graphical-capture tool is used to successively decompose a system into functional blocks defining the system's hardware architecture. Performance models for the leaf-level hardware blocks are defined by selecting an element from the PMW library, specifying the connections among its ports and the other elements in the system, and customizing the clement's parameters so that it exhibits the desired behavior. The software architecture of a system is defined by decomposing the software into tasks and specifying the messages communicated among each task. The behavior of each individual task is defined by creating a flow chart. The computational requirements of each flow chart block are specified by selecting predefined software elements from PMW library or by characterizing the instructions and memory operations executed by the block.

The PMW provides analysis and visualization support to enable a designer to verify the behavior of performance models, to identify bottlenecks and over-designs, and to compare design performance. Visualization support is necessary because of the vast amount of data communicated among the components of a complex system. This information overwhelms the displays provided by most VHDL simulators, especially when there is a large number of processors in a system.

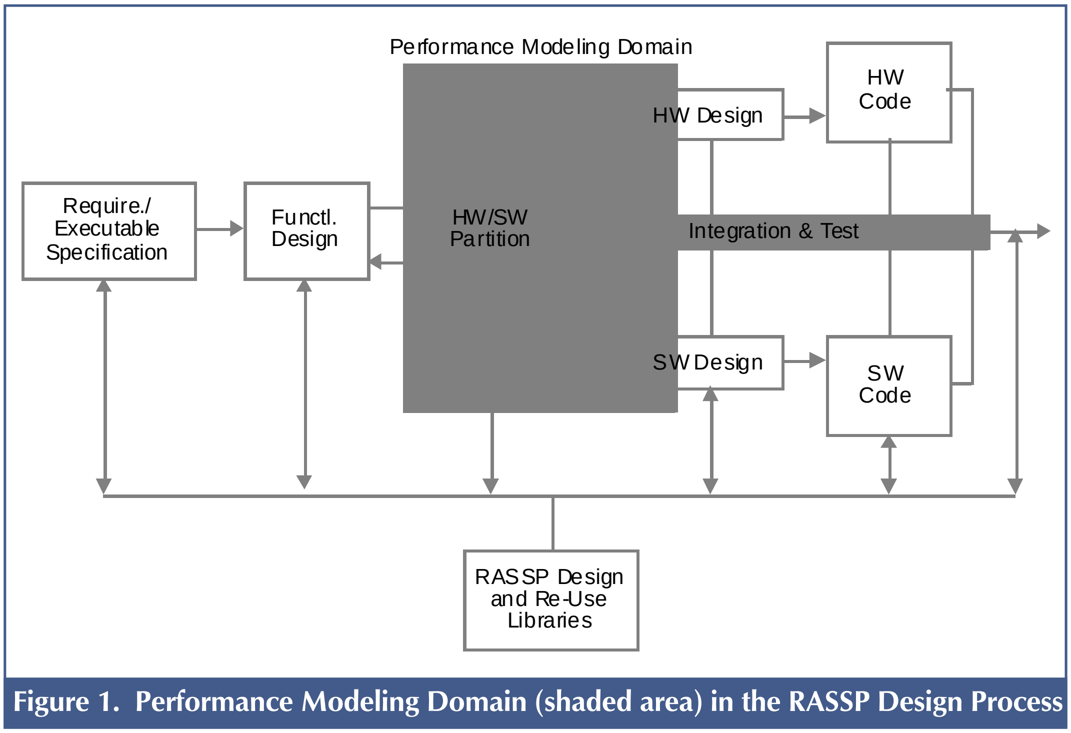

Performance Modeling Domain
HW Design
HW Code
Require./ Executable Specification
Functl. Design
HW/SW Partition
Integration & Test
SW Design
SW Code
RASSP Design and Re-Use Libraries

Figure 1. Performance Modeling Domain (shaded area) in the RASSP Design Process

The PMW analysis support includes standard analysis tools, such as activity timeliness bar graphs, and histograms, which display the latency, throughput, and utilization of system components (Figure 4). In addition, the PMW provides analysis tools that animate the architecture diagrams created in the hardware and software editors to display The system's simulation behavior. These animation tools color code the system components according to their latency, throughput, and utilization to help a designer quickly identify bottlenecks and overdesigns. The PMW can also export simulation results and performance metrics so that they can be loaded into spreadsheets or other analysis tools.

3. Conclusion

Prototypes of PMW are being used by RASSP participants, other DoD agencies and commercial companies. The feedback from these evaluation sites has been very positive and has been used by the development team to implement a continuous quality improvement process that will insure the commercial product resulting from the Technology Base research will meet their needs. A commercial product release is scheduled for the fourth quarter of 1996.

4. References

[1] Carpenter, T. & Hein, C., "VHDL-Based Top-Down Virtual Prototyping for Large DSP Systems, "Lockheed Martin Technical Presentation," Camden, NJ, November 2. 1996.

12] Rose, F., Steeves, T. & Carpenter, T., " VHDL Performance Models," *Proceedings 1st Annual RASSP Conference*, Arlington, VA, August 1994, pp 60-70.

[3] Steeves, T., et al, "Evaluating Distributed Muliprocessor Designs," *Proceedings 2nd Annual RASSP Conference*, Arlington, VA, July 1995, pp 95-102.

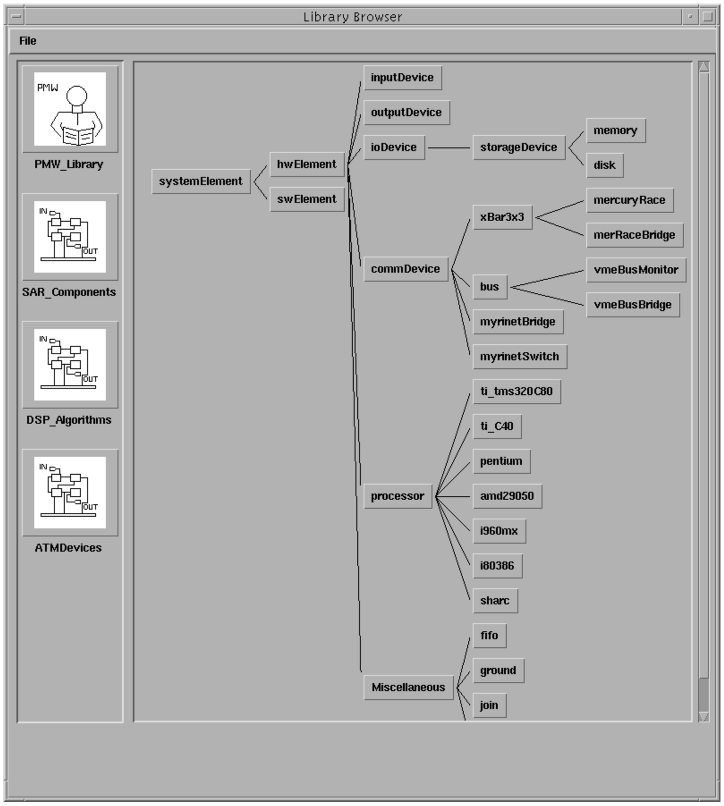


Figure 2. PMW's Performance Modeling Library

[4] Shackleton, J., & Steeves, T., “Advanced Multiprocessor Systems Modeling,” Proceedings Fall 1996 VIUF, Boston, MA, October 1995, pp 8.21-8.28.

[5] Carpenter, T., Rose, F., & Steeves, T., “VHDL-Architectural Assessment Environment.”

[6] Aylor, J.H., Waxman, R., Johnson, B.W., & Williams, R.D., “The Integration of Performance and Functional Modeling in VHDL,” Chapter 2 in *Performance and Fault Modeling with VHDL*, edited bv Joel M. Schoen, Prentice Hall, Inc., Englewood Cliffs, NJ,1992, pp 22-145.

Charles W. Buenzli, Jr. and Jay T. Runkel
Omniview, Inc.
100 High Tower Blvd., Suite 201
Pittsburgh, PA 15205
charles@omnivw.com, runkel@omnivw.com

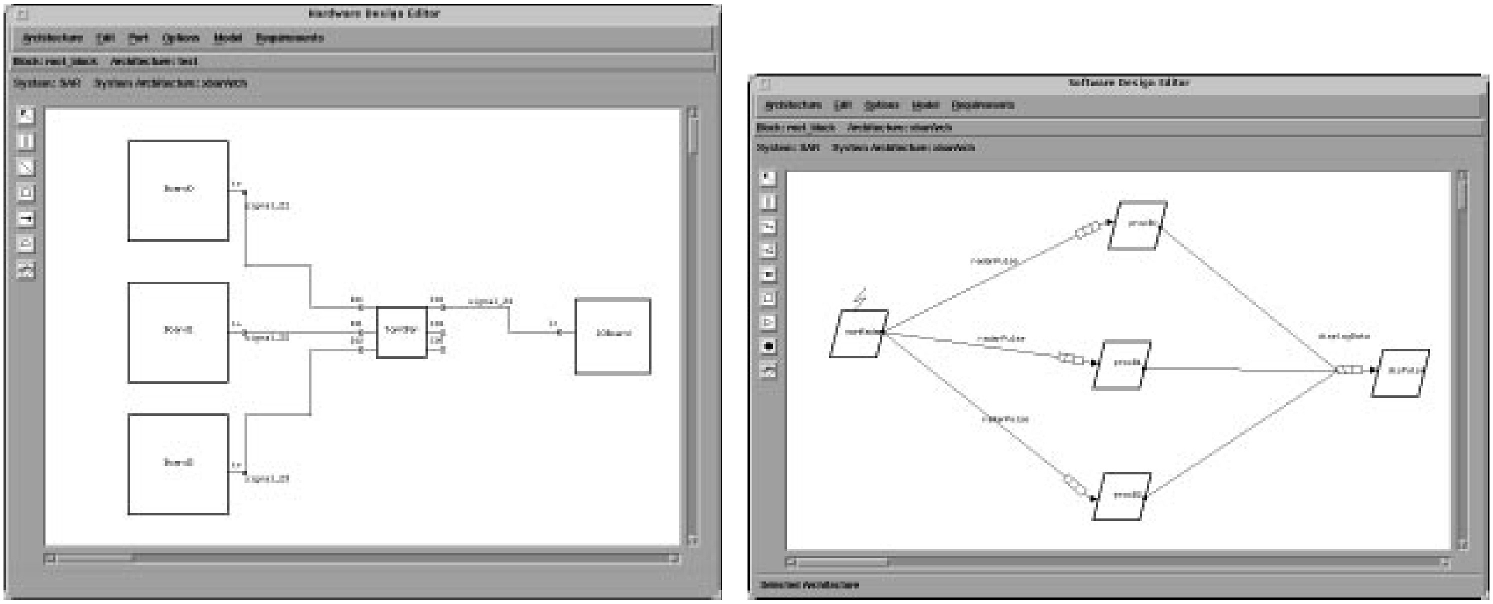
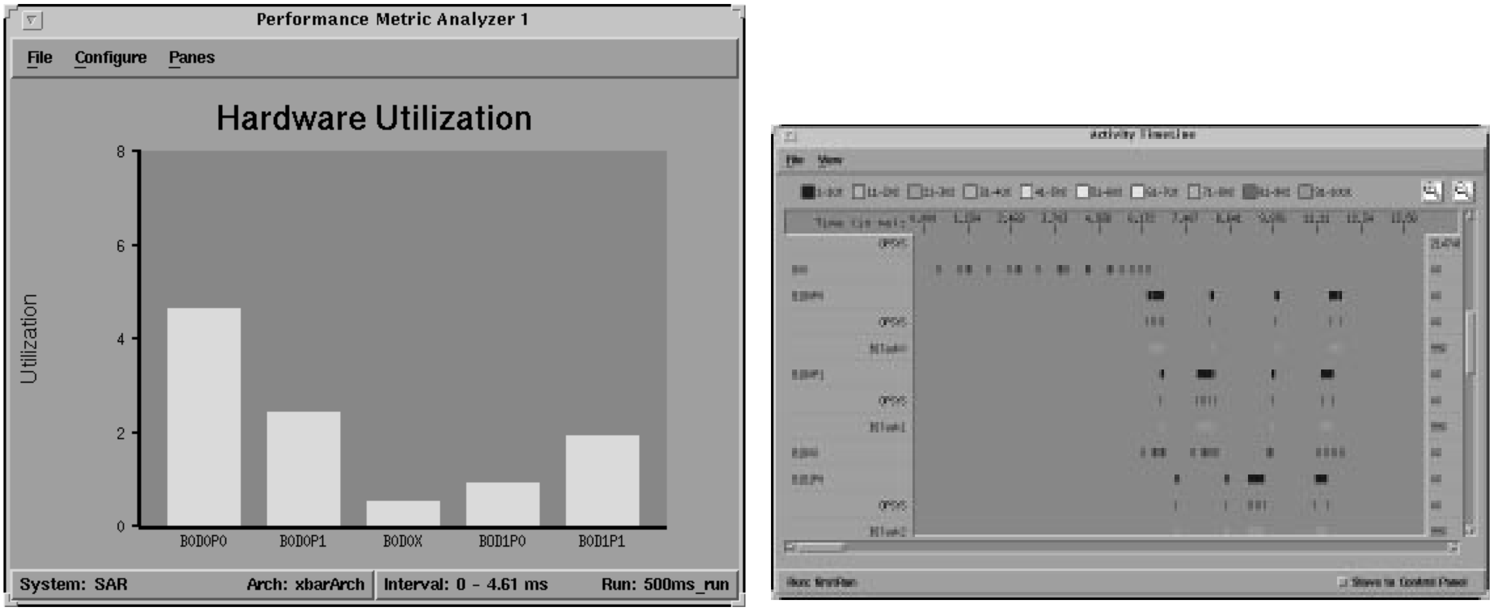


Figure 3. PMW Hardware and Software Design Editors



System: SAR

Arch: xbarArch

Interval: 0 – 4.61 ms

Run: 500ms_run

Figure 4. PMW's Analysis Tools

ANSI C to Behavioral VHDL Translator
Ada to Behavioral VHDL Translator

Robert J. Sheraga

1. Introduction

JRS Research Labs has developed two source language Translators, an Ada-to-VHDL Translator and a C-to-VHDL Translator, as part of the RASSP BAA program, under contract No. F33615-94-C-1497 with Wright Laboratory WL/AAKE.

The Translators perform source-to-source code translation, i.e., given an input program coded in Ada or C, the output is a behavioral VHDL source code program which is functionally equivalent to the original input program, in the sense that a VHDL compilation and simulation of the translated file will produce the same results as an Ada or C compilation and execution of the original source file.

Much of the program structure, data structures, identifier names, and even comments from the input program are preserved. The VHDL source file produced by the Translators is portable and independent of a specific VHDL implementation.

The Translators provide an automated method for translating existing code libraries from Ada or C into behaviors in VHDL which can be directly simulated. This provides a generalized tool to aid the process of hardware/software codesign, which can be used either standalone or incorporated into a larger tool framework.

2. Technical Approach

An intermediate form called the HIL (High-level Intermediate Language) was defined as an intermediate step in the translation process. The HIL form is at a relatively high level of abstraction, so that high level language constructs such as declarations, expressions, statements, etc. can be directly expressed. Compilers were developed using the Unix tools Lex (scanner) and Yacc (parser generator) to compile both Ada and C to the HIL form. A single HIL-to-VHDL Translator was then developed, which accepts the HIL form from both frontends and generates behavioral VHDL source code. This Translator module utilizes a rules file which defines the translation rules for each HIL statement. The user is permitted to provide certain types of special application-specific extensions to the translation rules. A runtime library was hand-coded in VHDL to provide some necessary runtime support for the translated programs.

3. Operation Summary

The input to the Translators is a source file coded in either Ada 95 or ANSI C. The full languages are supported for parsing, although some program constructs or statements may not be translatable into VHDL, and appropriate error or warning messages are issued for such statements. Ada input files may contain any number of compilation units. For C input files, the user has a choice of preprocessing header files or not.

The output program is structured such that any translated input file results in a VHDL output file which can be directly compiled in VHDL. Since VHDL does not permit subprograms as compilation units, procedures and functions which are translated are encapsulated in a package. The Ada Translator supports a rudimentary library system for context clauses, so that translating a set of Ada programs has the same order-of-translation requirements as compiling them with an Ada compiler, i.e., WITH'ed units must be translated first. For C programs, each input file is translated into a single VHDL package; statements outside functions are placed in the package specification, along with a subprogram declaration for each function. The package body contains the subprogram bodies.

The Translators can also automatically generate an entity/ architecture testbench for validating the translated programs, subject to some limitations.

4. Restrictions and Limitations

The Translators are, in general, limited to those features which are directly supportable in VHDL; however, there are some exceptions to this rule. Although the Ada-to-VHDL Translator accepts Ada 95 syntax, virtually all of the Ada 95 extensions, such as tagged, abstract, and protected types, child packages, etc., are not supported for VHDL translation, since there are no corresponding VHDL capabilities. The major features of Ada 83 which are unsupported are tasking, exceptions, subunits, fixed point types, representation clauses, most attributes, and certain classes of records not supported in VHDL, such as variant and discriminated records.

The C-to-VHDL Translator does not support variable length argument lists, abstract declarators, GOTO statements, automatic type conversions, certain types of complex switch statements, subarrays, and some other minor items. Because pointers are so important in C programs, partial support for pointers and pointer operations is provided even though VHDL does not support pointers directly. Support is limited to simple local pointers to scalar types. Thus, parameters or local variables declared as "int *" or "float *" are supported, but constructs such as pointers to functions, arrays, or structures, arrays of pointers, functions returning pointers, and pointers to pointers are not supported at this time.

A limited runtime support library is provided with the Translators, which includes primarily basic math routines. Specifically, the Translators do not include support for the full standard Ada 95 and ANSI C libraries. However, references to these library routines are not errors, per se, since they will result in the generation of valid VHDL source code containing equivalent calls. However, the user is responsible for creating the VHDL support packages necessary to compile and/or simulate programs which utilize these library functions.

5. Results and Status

The Ada-to-VHDL Translator correctly translates 302 of 315 Ada routines in a library of signal processing primitives obtained from the government. The C-to-VHDL Translator correctly translates 410 of 499 C routines in a commercial library of signal processing routines. These translations were validated by VHDL simulation of the translated source files. Numerous other test cases were also run successfully, including some standard Ada ACVC tests and standard C benchmark programs.

6. Summary and Conclusions

Hardware and software development are no longer separate camps, but are fast merging as a result of hardware/software codesign methodologies utilizing tools which focus on the overall system design problem. As the boundary between hardware design and software design continues to blur, it becomes increasingly important that algorithms be transportable, or at least easily convertible, between functionality expressed in Ada and C on the software side, and VHDL on the hardware side. If we view Ada and C not only as programming languages but also as specification languages for algorithms for which we intend to eventually synthesize hardware implementations, then the importance of these issues becomes even more apparent. A VHDL version of an algorithm coded in Ada or C can be used to synthesize custom hardware to implement the algorithm, or as part of a VHDL functional simulation of a processor designed to execute the algorithm.

Future considerations for the Translators include removing some of the language restrictions, generating optimized VHDL code, and generating VHDL code which is optimized for synthesis and/or targeted to a specific synthesis environment.

7. Example Program

Input file:

```
------------
/* Complex Vector Power Computation -- C version */

void voc_pwr_c (n, x, y, z)
int    n;
float  *x;
float  *y;
float  *z;
{
        int i;
        float *xp, *yp, re, im;
        xp = x;
        yp = y;
        *z = 0.0;
        for (i=0; i < n; i++) {
                re = *xp++;
                im = *xp++;
                *yp = re*re + im*im;
                *z += *yp++;
        }
}
```

Output file:

```
--------------
use work.C_Data_Types.all;
use work.HVT_Runtime_Library.all;
package voc_pwr_c is
        -- Source file is /a+b/sonia/hvt/c2h/src/voc_pwr_c.c
        -- Source language is C
        --/* Complex Vector Power Computation -- C version */
        procedure voc_pwr_c
                (n : in Integer;
                x_Array : inout Float_Array;
                y_Array : inout Float_Array;
                z_Array : inout Float_Array);
end voc_pwr_c;
package body voc_pwr_c is
        procedure voc_pwr_c
                        (n : in Integer;
                        x_Array : inout Float_Array;
                        y_Array : inout Float_Array;
                        z_Array : inout Float_Array) is
                variable x : Pointer_to_Float := x_Array'Left;
                variable y : Pointer_to_Float := y_Array'Left;
                variable z : Pointer_to_Float := z_Array'Left;
                variable i : Integer;
                variable xp : Pointer_to_Float := nil;
                variable yp : Pointer_to_Float := nil;
                variable re : Real;
                variable im : Real;
        begin
                xp := x;
                yp := y;
                z_Array(z) := 0.0;
                i := 0;
                while i < n loop
                        re := x_Array(xp);
                        xp := xp + 1;
                        im := x_Array(xp);
                        xp := xp + 1;
                        y_Array(yp) := re * re + im * im;
                        z_Array(z) := z_Array(z) +
                        y_Array(yp);
                        yp := yp + 1;
                        i := i + 1;
                end loop;
        end voc_pwr_c;
end voc_pwr_c;
```

Robert J. Sheraga
JRS Research Laboratories, Inc.
1036 W. Taft Avenue
Orange, California   92665-4121
bob@jrs.com

MAT2DSP - A MATLAB Tool for the Automatic Evaluation of the Implementation Requirements of Signal Processing Algorithms

S. Bose and B. Friedlander

Abstract

MAT2DSP is a MATLAB toolbox whose function is to estimate the implementation requirements of advanced signal and image processing algorithms specified in the form of MATLAB programs.

1. Objectives

The MAT2DSP toolbox seeks to be a user-friendly tool designed for researchers developing advanced signal and image processing algorithms. This is intended to supply quick and rough estimates of what would be needed to implement their algorithm on a specified processor (or processors) without necessarily involving them in all the implementation details. Such rapid feedback will enable the researcher to do a rudimentary but painless cost/benefit analysis early in the design process and focus on algorithms which are likely to meet the system constraints.

2. Summary of Technical Approach

The main approach being pursued here is to decompose any algorithm into its basic functional units called primitives and use a database containing the cost/requirements for implementing each primitive on any specified processor to obtain an estimate for the whole algorithm or any of its functional parts on that processor. Such an estimate is necessarily rough since it ignores the possibility of optimization in the implementation. However this can be supplied quickly and easily without requiring the target user to get involved with all the finer aspects of the processor implementation, enabling a rapid and easy feedback as stated in the objective.

The MAT2DSP has four main components - a program analyzer, a database incorporating information on various DSP processors, a report generator for presenting the estimates of cost/requirements and a database creation package for generating the database for each DSP processor.

The program analyzer's task is to parse the input MATLAB program and generate a functional profile of the algorithm, i.e., break up the algorithm in a hierarchy of functional sub-units terminating at the primitives. Further, the costs associated with each sub-unit are obtained in one of two alternative modes -- pre-run analysis and run-mode analysis. This is done using a translated version of the target program in which all the primitives are mediated via a special program which therefore allows access to them. The pre-run analysis tool generates the estimates without actually running the algorithm; the estimates are then independent of the actual values of the input. The run-mode analysis tool, invoked when pre-run analysis is not possible, obtains the same estimates while running the target program, primitive by primitive.

The report generator combines the information in the functional profile along with the database information to present reports summarizing or detailing the cost/requirements to the user. A user interface is intended to enable the user to customize the report and make comparisons across different processor platforms.

The database creation package is used initially to generate the database and subsequently to update or augment it. It involves an experiment manager to benchmark implementations of the primitives on different platforms and a database generator to compile and consolidate the collected data from the experiment manager into the database for use by the report generator.

Figure 1 gives an overview of the whole MAT2DSP package highlighting the interrelationship of its main components.

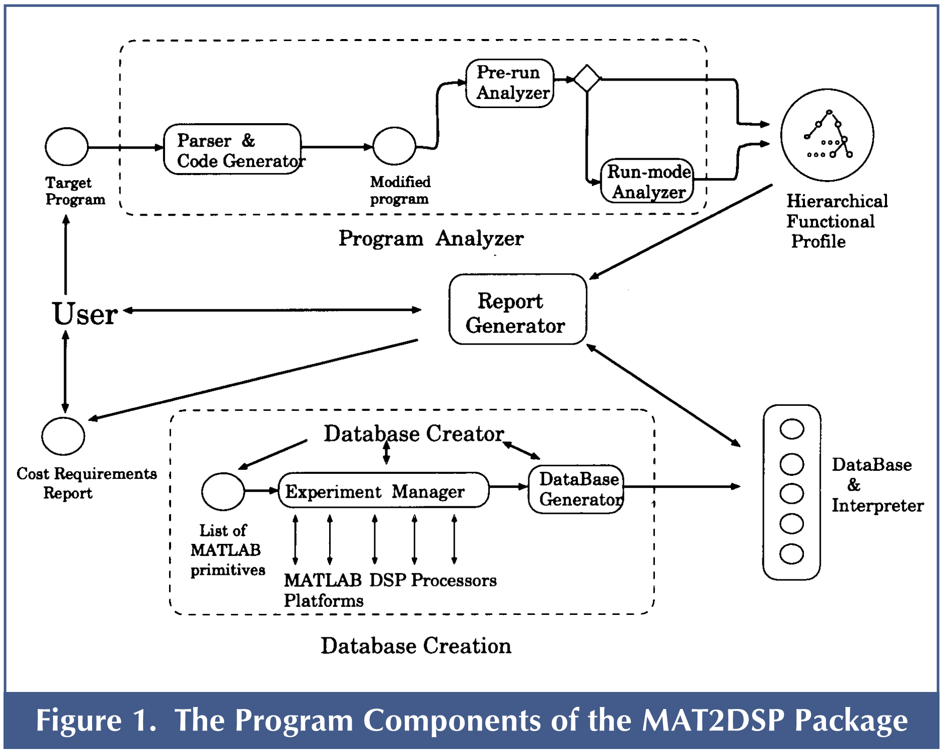
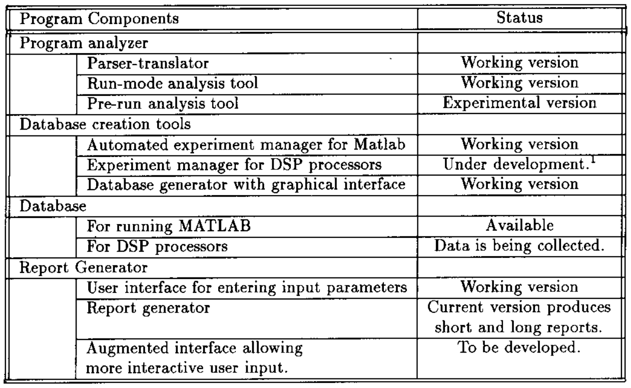


Figure 1. The Program Components of the MAT2DSP Package

3. Technical Contributions/Achievements

As of now the program analyzer component has been completed and a version is being tested. An automated experiment manager has been developed for obtaining data running MATLAB on various platforms. However the data for DSP processors has to be collected manually which is the task currently underway. A simple report generator and user interface has also been developed.

Table 1 summarizes the current status of the various components.



Program Components		Status
Program analyzer		
	Parser-translator	Working version
	Run-mode analysis tool	Working version
	Pre-run analysis tool	Experimental version
Database creation tools		
	Automated experiment manager for Matlab	Working version
	Experiment manager for DSP processors	Under development.[1]
	Database generator with graphical interface	Working version
Database		
	For running MATLAB	Available
	For DSP processors	Data is being collected.
Report Generator		
	User interface for entering input parameters	Working version
	Report generator	Current version produces short and long reports.
	Augmented interface allowing more interactive user input.	To be developed.

Table 1. Current Status of the MAT2DSP Tools

As of now, it has been possible to compare estimates of the running time of MATLAB programs with the actual times. The observed discrepancy of 20-50 is well within the design goals.

4. Deliverables and Availability

At the end of the project the MAT2DSP toolbox and related documentation will be delivered to the government.

5. Publications

[1] B. Friedlander, "MAT2DSP -- A Tool for Evaluating Implementation Complexity of Signal Processing Algorithms," Intl. Conference on Acoustics, Speech and Signal Processing, Detroit, Michigan, 5/8-12/95.

[2] A. Zeira, S. Bose and B. Friedlander, "MAT2DSP - A Tool for Automatic Evaluation of the Implementation Costs of Signal Processing Algorithms," submitted to the Journal of VLSI Signal Processing, special issue on "Design and Implementation of Signal Processing Systems."

[3] A. Zeira and B. Friedlander, "Run-time analysis of the computational requirements of MATLAB programs," MAT2DSP Technical Report, July 1995.

[4] S. Bose and B. Friedlander, "The MAT2DSP Database Generator," MAT2DSP Technical Report, September 1995.

S. Bose and B. Friedlander
Department of Electrical and Computer Engineering
University of California
Davis, CA 95616
sbose@ece.ucdavis.edu, friedlan@ece.ucdavis.edu

Timing Insensitive Binary-to-Binary Translation (TIBBIT)

Bryce Cogswell and Zary Segall

Abstract

TIBBIT provides technologies that automate the migration of legacy embedded software systems to model-year platforms while preserving their semantic and timing properties. A working demonstration system is complete and has been modeled and validated for software running on a number of source and target architectures.

1. Objectives

The majority of software for embedded systems is developed for a specific processor and platform environment. In addition to being targeted toward a specific instruction set architecture (ISA), such software typically incorporates hard-coded knowledge of the system's I/O ports, timing facilities, interrupt mechanism, etc. In many instances, the code may also take advantage of the programmer's knowledge of processor timing information, i.e., either the minimum or maximum time a given code fragment may require to execute. Upgrading a legacy system designed these assumptions to model year hardware typically requires a complete rewrite of the software, foremost to achieve ISA compatibility, but also because locating and correcting every instance of reliance upon the original specification is a daunting task. The TIBBIT project aims to provide an automated migration path for software developed on legacy hardware to model year hardware. Instances of program reliance upon source-hardware features are automatically detected and modified to reflect the target-hardware environment.

The goals of the TIBBIT methodology are as follows:

- *Semantic equivalence:* The resultant program is semantically equivalent to the original program.
- *External timing equivalence:* The timing of the program on the target is equivalent to the source platform within some predictable error bound.
- *Processor independence:* The scheme should be effective across a wide range of processor architectures.
- *Use existent I/O architecture:* The interfaces to external devices to which the source processor is attached are preserved.
- *Quantifiable performance:* The success of a translation can be predicted prior to translation, and the degree of timing equivalence can be quantified.

- *Automated translation:* The translation process should be nearly or entirely automated.

2. Technical Approach

TIBBIT builds upon previous work in binary-to-binary translation (BBT), where the object code of an application is analyzed using control and dataflow analysis, and instruction sequences (e.g., basic blocks) are then translated from the source ISA to the target ISA, preserving the semantics of the program. TIBBIT extends this process by determining not only the semantic properties of the program, but also the timing properties. During translation the time required to execute each basic block is computed for the source ISA, and this is used to modulate the speed at which the application executes on the target ISA. At regular intervals the target processor compares its progress with what would be achieved by the source ISA, and adjusts the amount of processing time dedicated to the task accordingly. Scheduling on the target can be done two different ways: to maximize the degree of timing equivalence a dedicated target processor is used; otherwise, the application is scheduled on the target under rate monotonic scheduling. Using RMS allows multiple TIBBIT- translated and native applications to be executed concurrently.

To facilitate the creation of binary translators for new architectures, an automated translator generator, called *Astra,* has been created. Astra takes as input a machine description file which specifies the syntax and semantics of a source ISA's binary format, and produces a binary translator capable of decompiling applications into an intermediate representation that is subsequently compiled using gcc. This translation approach eases ports to new source ISA's and yields target ISA independence.

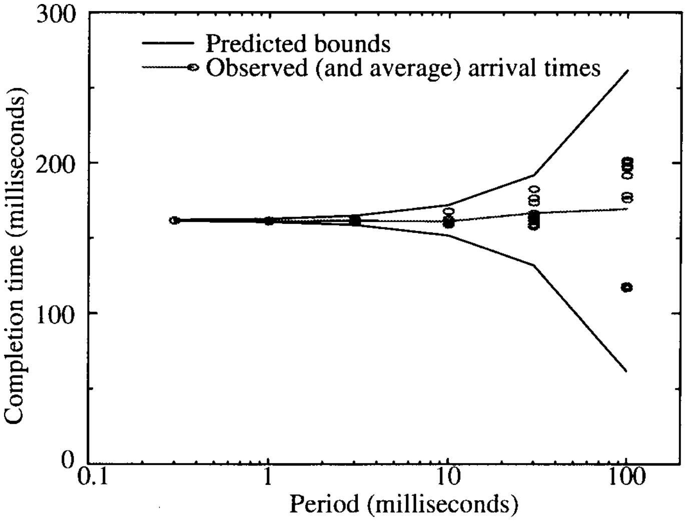
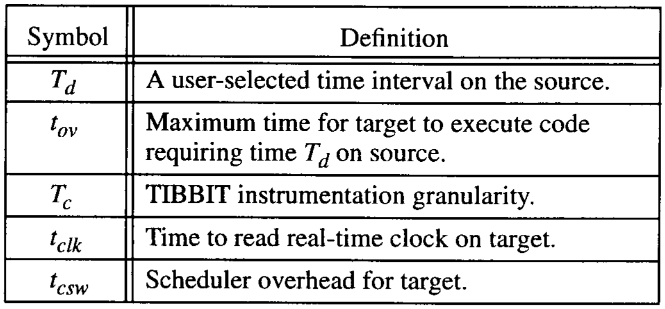


Figure 1. Timing Error as a Function of Scheduled Period

3. Technical Contributions

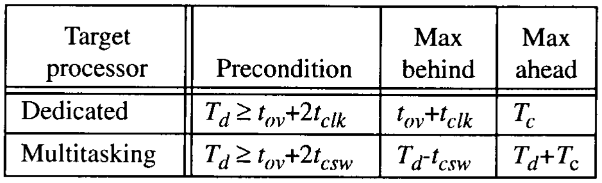
The TIBBIT project has yielded a methodology and prototype translation system capable of migrating applications from legacy to model-year hardware while maintaining all semantic and timing characteristics. Sample translators have migrated applications from the M68000 and TMS320C30 architectures to MIPS, PowerPC, Sparc and Pentium based systems. Translated systems retain timing equivalence with the original systems to within 80 microseconds while scheduled under the Rate Monotonic Scheduling algorithm on the target with other tasks (Figure 1), and within 25 microseconds when executing on a dedicated processor. Processor utilization overhead incurred by the timing instrumentation is about 20%.

A model predicting the degree of timing equivalence achievable for a given translation has been developed and validated. The model uses characteristics of the source and target ISA timing, as well as the application being translated, to compute the timing accuracy that can be achieved by the translation software. Current work builds on this foundation to further reduce timing error by reordering instructions during the translation to minimize "trouble spots" in the code.



Symbol	Definition
T_d	A user-selected time interval on the source.
t_{ov}	Maximum time for target to execute code requiring time T_d on source.
T_c	TIBBIT instrumentation granularity.
t_{clk}	Time to read real-time clock on target.
t_{csw}	Scheduler overhead for target.

Table 1. Summary of TIBBIT Model Parameters



Target processor	Precondition	Max behind	Max ahead
Dedicated	$T_d \geq t_{ov}+2t_{clk}$	$t_{ov}+t_{clk}$	T_c
Multitasking	$T_d \geq t_{ov}+2t_{csw}$	T_d-t_{csw}	T_d+T_c

Table 2. Summary of TIBBIT Algorithm Characteristics

Table 1 specifies the factors affecting TIBBIT schedulability while Table 2 specifies the greatest amount by which timing on the target processor can become out of sync with the source. The precondition column specifies the condition that must hold for the task to be schedulable under TIBBIT, while the max behind and max ahead columns bound the maximum timing error that can occur.

4. Deliverables

The following deliverables have been produced in the course of the TIBBIT project. Basic methods and techniques for semantic and timing equivalent legacy migration were developed. A working implementation used to translate monolithic applications and executives on uniprocessor systems is complete, and has been validated using real-world programs. The retargetability of the system has been demonstrated on numerous platforms.

Finally, the Astra translator generator has been demonstrated for translating from both CISC and DSP architectures to various RISC platforms.

5. Technical Papers

Further details on the implementation and results of the TIBBIT project are available in a number of technical papers. The fundamental concepts of the system are first proposed in “Timing Insensitive Binary to Binary Translation of Real Time Systems” (Cogswell and Segall, Carnegie Mellon Tech Report, March ’93). Initial results for the timing equivalence implementation are described in “Timing Insensitive Binary to Binary Translation” (Cogswell and Segall, Real-time workshop ISCA ’94). “Performance Impact of Architectural Features During Binary Translation” (Cogswell and Segall, PACT ’95) analyses the architectural features in the source and target ISAs that contribute to overhead incurred during the translation process, quantifying the features that make a target ISA a good “match” for a legacy system. The feasibility and limitations of supporting multiprocess architectures, as either a source or target platform, are examined in “Timing Insensitive Binary-to-Binary Migration Across Multiprocessor Architectures” (Cogswell and Segall, WPDRTS, ’95). Finally, a comprehensive look at the modeling, implementation and results of the project is provided in “Timing Insensitive Binary to Binary Translation” (Cogswell, Carnegie Mellon Ph.D. thesis, ’95).

Bryce Cogswell and Zary Segall
Computer and Information Science
University of Oregon
Eugene, OR 97403
zs@cs.uoregon.edu, cogswell@cs.uoregon.edu

Design Tools and Architectures for Dedicated DSP Processors

Keshab K. Parhi and Ching-Yi Wang

Abstract

Current demands on prototyping complex DSP applications has placed more emphasis on developing efficient methods and CAD tools. In our research, we have developed the Minnesota ARchitecture Synthesis (MARS) system that is capable of converting high-level behavioral descriptions into hardware architectures in very short run times. MARS utilizes the Mentor Graphics toolset as a GUI, simulator base, and layout editor. MARS also provides a structural VHDL output of the final design. Other efforts have developed novel and efficient architectures for difficult DSP applications such as: discrete wavelet transforms, novel topologies for ADSL/HDSL applications, and Reed-Solomon encoders.

1. Introduction

Our primary goals for the RASSP project is to develop CAD tools that efficiently explore various design decisions and their impact on the final implementation. We are also investigating efficient and novel architectures for difficult signal processing applications. The motivation behind developing these new architectures is to gain a better understanding of efficient translations from algorithm to hardware and this will lead to better CAD tools. We have been developing the Minnesota ARchitecture Synthesis (MARS) system which will automatically generate high-performance, dedicated architectures within a heterogeneous design environment. We are also investigating the design of high-performance and low-power architectures for the following DSP applications: discrete wavelet transforms, novel topologies for ADSL/HDSL applications, finite field arithmetic architectures, and Reed-Solomon encoders.

2. Design Tools

In the process of rapidly designing prototypes of real-time DSP systems, the use of high-level synthesis has become a more common and crucial step in the design flow because many real-time applications require high sample rates or low power consumption that can only be implemented by dedicated architectures. We have developed two approaches to this problem, a heuristic technique and an optimal integer linear programming (ILP) model based technique. Heuristic methods are attractive because they can generate good results in short CPU times; however they cannot guarantee optimal solutions. The more formalized approaches that utilize ILP models are attractive because they are capable of generating optimal solutions and are more flexible; however they suffer in exponential increases in run times as the design constraints become less restrictive. Most of the previously developed synthesis systems assume that all same type operations will be assigned to one type of hardware functional unit (e.g., all addition operations will be processed by full adders). A few systems allow for different types of processors for the same type operations; however, they only utilize homogeneous architectures where all of the processors are implemented using a single implementation style such as bit-parallel or bit-serial. Our work considers synthesis using a heterogeneous architecture environment where different types of functional units (including implementation styles) can be used to process same type operations. By allowing heterogeneous

This research was supported by the Advanced Research Projects Agency and the Solid State Electronics Directorate, Wright-Patterson AFB under contract number AF/F33615-93-C-1309.

processors in the final architecture, the data format of one processor may not necessarily be the same as another processor [1][2][3]. For example, the final design may contain an adder which computes one word in one clock cycle and a second adder which processes a half-word in one clock cycle. This leads to the need for data-format converters which accept input data in a certain format and generate output data in a different format where the data format may be bit-serial or digit-serial or bit-parallel.

3. Architectures

We are also investigating the design of efficient and novel architectures for three different and unrelated DSP applications. Through this work, we have developed new architectures and systematic techniques that have lead to more robust CAD tools. The three different DSP applications that we are pursing include high-performance architectures for the discrete wavelet transform, new designs for ADSL/HDSL applications, more efficient architectures for finite field arithmetic, and low-power Reed-Solomon encoders. We developed new systematic techniques for designing efficient lattice structure discrete wavelet transform architectures. From this work, we described a general approach to design efficient architectures for multirate applications [4][5]. We have also developed novel topologies for ADSL/HDSL applications [6]. In this work we compared the efficiency of the various architectures and analyzed the tradeoffs during the design process. We developed and implemented new low-latency arithmetic processors for finite field applications and designed efficient Reed-Solomon encoders [7].

4. Deliverables

Our deliverables include the MARS synthesis toolset, a small test library of functional units and converters, and all publications on all aspects of our research supported by the RASSP contract.

5. Conclusion

We developed two techniques that efficiently performs high-level synthesis within a heterogeneous design environment. We developed a heuristic that is fast and efficient and a set of ILP models that are more flexible and can guarantee optimal solutions. Both techniques generate a low-cost solution (including data-format converters) in terms of total area consumed. This heuristic algorithm has been incorporated into the MARS-II synthesis system and is integrated within the Mentor Graphics toolset. Although the heuristic approach cannot guarantee optimal results, our experiments have shown that MARS-II is able to produce optimal and near optimal solutions in one to two orders of magnitude less time than our more robust ILP models. We have developed new systematic techniques for designing efficient discrete wavelet transform architectures. We have also developed new architectures for ADSL/HDSL applications and new low-latency arithmetic processors for finite field applications and low-power Reed-Solomon encoders.

References

[1] K. Ito, L. E. Lucke, and K.K. Parhi, “Module Selection and Data Format Conversion for Cost-Optimal DSP Synthesis,” in *International Conference on Computer Aided Design*, (San Jose, CA), pp. 322-327, November 1994.

[2] C.-Y. Wang, and K.K. Parhi, “High-Level DSP Synthesis using Concurrent Transformations, Scheduling, and Allocation,” *IEEE Transactions on Computer Aided Design*, Vol. 14, No. 3, pp. 274-295, March 1995.

[3] Y.-N. Chang, C.-Y. Wang, and K.K. Parhi, “Loop-List Scheduling with Heterogeneous Functional Units,” in *Great Lakes Symposium on VLSI*, (Ames, IA), pp. 2-7, March 1996.

[4] T. C. Denk and K. K. Parhi, “Systematic Design of Architectures for M-ary Tree-Structured Filter Banks,” in *VLSI Signal Processing VIII* (T.Nishitani and K. K. Parhi, eds.), pp. 157-166, IEEE Press, 1995. (Proc. of the 1995 IEEE Workshop on VLSI Signal Processing, Osaka, Japan).

[5] T. C. Denk, M. Majumdar, and K. K. Parhi, “Two-Dimensional Retiming with Low Memory Requirements,” in *International Conference on Acoustics, Speech, and Signal Processing*, (Atlanta, GA), pp. 3330-3333, May 1996.

[6] A. Shalash and K. K. Parhi, “Discrete Mutlitone Versus Carrierless AM/PM Architecture Comparison,” in *International Symposium on Circuits and Systems*, (Atlanta, GA), pp. II:560-564, May 1996.

[7] S. K. Jain and K. K. Parhi, “Efficient Standard Basis Reed-Solomon Encoder,” in *International Conference on Acoustics, Speech and Signal Processing*, (Atlanta, GA), pp. 3287-3290, May 1996.

Keshab K. Parhi and Ching-Yi Wang
Department of Electrical Engineering
University of Minnesota
200 Union Street SE
Minneapolis, MN 55455
parhi@ee.umn.edu

VHDL Hybrid Models

Fred Rose

Abstract

Hybrid modeling is the capability of mixing high-level performance constructs and functional components in a common analysis environment. A coordinated research effort between the Honeywell Technology Center (HTC) and the Center for Semicustom Integrated Systems at the University of Virginia (UVa) is addressing issues related to hybrid modeling.

1. Introduction

The primary objective of hybrid modeling is to handle the complex task of translating data and control flow between models at different levels of abstraction and interpretation. The necessary information content at interface boundaries varies with the model abstraction level. As models become more detailed the amount of information in the interface increases. Differences in information content cannot be handled purely via translation (such as translating an integer representation of data into a bit vector). Rather, information might need to be synthesized or otherwise generated when interfacing to a more detailed model. When interfacing to a less detailed model, information might need to be discarded, encapsulated, or abstracted. This must all occur seamlessly within a single model context, where multilevel models interconnected with the hybrid interface components operate in a integrated fashion.

Within the RASSP methodology, hybrid modeling supports the spawning of mini-spirals to develop critical or high risk items. Based upon risk, pieces of the overall design may be at differing maturity levels. This is where the concept of hybrid modeling becomes important. As the overall system model is developed, certain high risk areas need further development. From a modeling perspective, this will be accomplished with more detailed models. Using modeling terminology, the system level uninterpreted (performance) model will have certain components replaced with the more detailed, interpreted (functional) models. To continue the RASSP philosophy of virtual prototypes, methods must enable the mixture of different model abstractions.

The hybrid modeling capability permits incorporation of detailed, functional or behavioral models into the overall system performance model. This allows the designer to explore and determine effects of the detailed design which would not otherwise be well characterized or understood. The RASSP concept of model/design reuse means that major parts of a new design will be constructed from well understood components, where these components may be processors, MCMs, or modules. Therefore a good characterization of these components should exist for use in uninterpreted modeling. However, characterizations for high risk or new components are usually initial approximations. As that new component design becomes more refined, these initial approximation may change. That change could have a ripple effect throughout the entire design. Hybrid modeling permits heterogeneous simulation and analysis to occur within the VHDL simulation environment to establish good estimates of behavior early in the design cycle.

The hybrid model allows validation of certain assumptions and dependencies. While a detailed model of the critical portion might run stand-alone, and the timing results back-annotated into the performance model, this may not work in all cases. For instance, many classes of contention, such as memory, network, and processor scheduling and overhead, might only be resolved with the majority of the system modeled, providing accurate workload characterizations. Hybrid models permit this early in the design cycle, without requiring the entire system be modeled at an equivalently detailed level.

2. Approach

The approach to the hybrid modeling problem has been to define a taxonomy for hybrid models, develop a common structure for hybrid architectures, and develop a set of generic VHDL architectures and library elements to support a wide range of hybrid modeling applications.

The classes of hybrid modeling are defined by those model attributes which fundamentally alter the development and implementation of the hybrid interface. The hybrid modeling space is partitioned according to the following characteristics:

1. The hybrid model *objective*
2. The timing and *synchronization mechanism* of the model
3. The *nature* of the interpreted element
4. The data transformation *mode*
5. The *data type* of the interpreted signals

The details of the hybrid taxonomy are contained in [1].

Honeywell has developed a library of hybrid architectures to enhance our existing VHDL Performance Modeling Library (PML) [2]. The commercialization of the PML is discussed in an accompanying article.

The typical hybrid architecture is shown in Figure 1. As with the PML, an attempt was made to design the architectures as generic as possible. Unfortunately this is more difficult for hybrid architectures than for performance architectures. The functional, or interpreted, component interface can be significantly different from one model to the next. Performance models, on the other hand, typically use a similar signal structure.

The approach of developing hybrid architectures for existing PML components was chosen for several reasons. The insertion of functional model components into a performance model is a likely methodology step under the RASSP methodology. Also the existing PML gives a solid basis for developing an initial library of hybrid architectures for validation purposes. HTC is concentrating on the implementation aspects of hybrid models while UVa is focusing on some specific research issues [1] such as models with sequential interpreted elements with known and unknown inputs, and adding hybrid capability to the ADEPT environment. Additionally, UVa has developed a set of components to allow the integration of the UVa ADEPT models and HTC PML models within one simulation environment.

3. Results

Releases of the Hybrid Modeling Library (HML) have been made in December 1995, and May 1996. The HML is released in conjunction with the PML. The HML contains hybrid architectures for the following PML components: indevice, outdevice, iodevice, pipeline, processor, and a communication element. The HML also contains the PML/ADEPT interface components. Examples of interpreted components are also contained in the HML to provide examples of complete hybrid architectures. The May release of the library also contains a hybrid interface generation toolkit. The purpose is to make the development of hybrid architectures as easy as possible for the user. Additionally the HML will be integrated with the commercial PMW and other capabilities of the Omniview tools are being investigated with respect to hybrid architecture generation.

Verification of the HML against real world design examples is ongoing. Results of this work will be available at a later date.

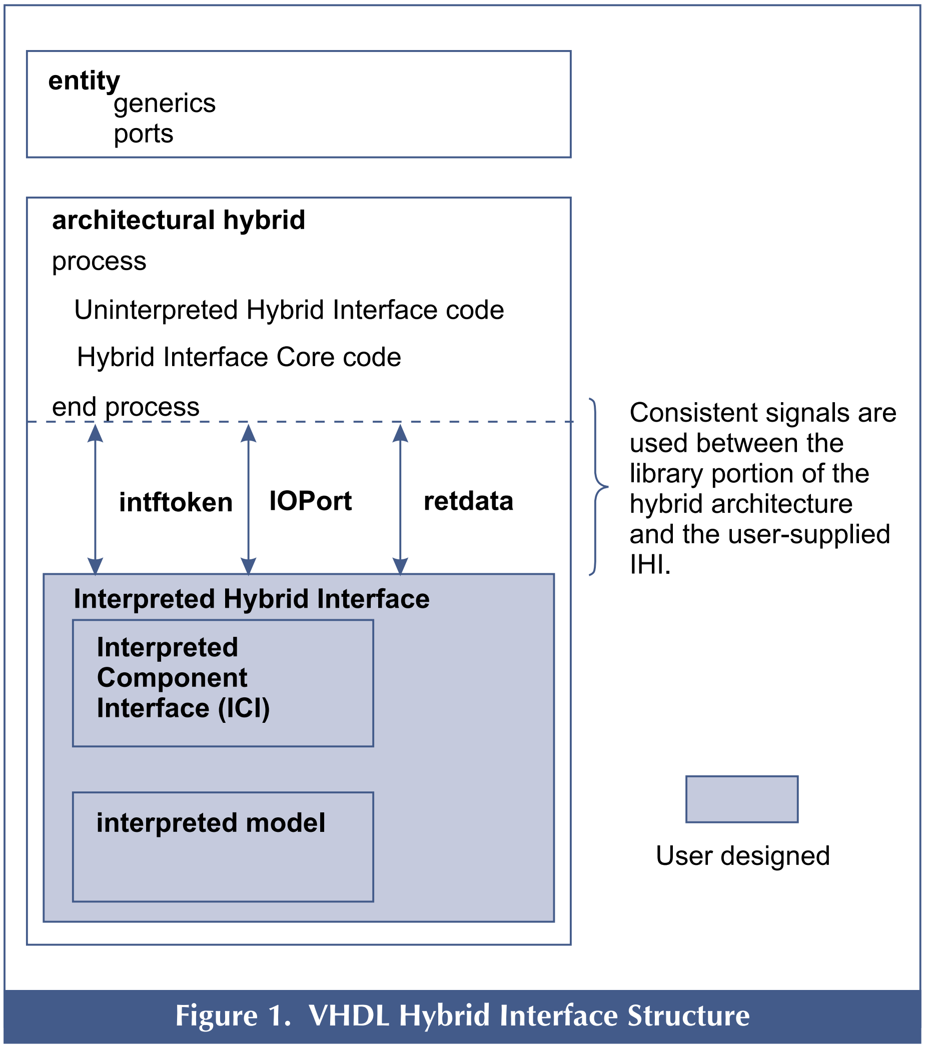

entity
generics
ports
architectural hybrid
process
Uninterpreted Hybrid Interface code
Hybrid Interface Core code
end process
intftoken
IOPort
retdata
Consistent signals are used between the library portion of the hybrid architecture and the user-supplied IHI.
Interpreted Hybrid Interface
Interpreted Component Interface (ICI)
interpreted model
User designed

Figure 1. VHDL Hybrid Interface Structure

The PML, and associated HML, are available to RASSP contractors through Omniview. They will both be available through the commercial Performance Modeling Workbench.

For more information about the PML and ADEPT, reference the HTC (*http://www.htc.honeywell.com/projects/rassp/*) and UVa (*http://csis.eee.virginia.edu/rassp/adept.html*) Web pages, which are available off the E&F Web page (*http://rassp.scra.org/*).

References

[1] Meyassed, M., et al, "A Framework for the Development of Hybrid Models," RASSP Conference Proceedings, Washington, D.C., July, 1995.

[2] Steeves, T., F. Rose, T. Carpenter, J. Shackleton, O.von der Hoff, "Evaluating Distributed Multiprocessor Designs,'' RASSP Conference Proceedings, Washington, D.C., July, 1995.

Fred Rose
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
rose_fred@htc.honeywell.com

Automated Generation of VHDL Processor Models for Simulation and Synthesis

Vijay K. Madisetti and Yong-Kyu Jung

Abstract

A new process for automating the creation of full-behavioral and Instruction Set Architecture (ISA) models in VHDL for complex processors and components is described, with results from the automation of a PowerPC 601 described in some detail. A number of advantages to this approach are described together with its impact on the hardware/software codesign and system prototyping processes.[1]

1. Introduction

The Rapid Prototyping of Application-Specific Signal Processors (RASSP) project of the US Department of Defense (ARPA and Tri-Services) targets a 4X improvement in the design, prototyping, manufacturing, and support processes (relative to current practice). As per current practice (circa 1993), the prototyping time from system requirements definition to production and deployment, of multiboard signal processors, is between 37 and 73 months [8]. Out of this time, 25-49 months is devoted to detailed hardware/software (HW/SW) design and integration (with 10-24 months devoted to the latter task of integration). With the utilization of a promising top-down hardware-less codesign methodology based on full-behavioral VHDL models of HW/SW components, it appears feasible that the HW/SW integration time could be reduced to a few weeks (1-2 months) [10]. Potential show-stoppers lie in the limited availability of high quality VHDL behavioral models of components (timing and function). In addition, the time to build a single model of a complex RISC processor (such as i860XP) is approximately a person-year. We describe a mechanism via which full-behavioral models of complex components can be automatically generated in VHDL from published information available in data manuals. This method could also be used in the iterative design synthesis of custom pipelined processors for domain/application-specific applications.

2. HW/SW Codesign Practice

The Educator/Facilitator current practice (1993) model for signal processor design is presented in detail in [8] in this proceedings. The various stages in a "waterfall"-type process flow are demarcated together with time ranges (min, max) for each stage. The time lines have also been validated via communications with the industrial entities involved in large system design and implementations. In this paper, we focus on the specific tasks of hardware (HW), software (SW), and interface design and their eventual integration.

2.1 Whither True Codesign?

True HW/SW codesign allows both hardware and software to be designed within a common framework, and simulated together before being fabricated. Current practice attempts to automate this process via HW/SW/Interface partitioning followed by three individual paths to HW, SW and Interface design and implementation, respectively (as shown in Figure 1). A drawback with this approach is that software can be designed and tested only if a hardware platform (at board and rack levels) is available. The latter is time- and cost-consuming (even if it utilized FPGA technology or HW modelers). It must be understood that the software is not just application-specific software, but also control, diagnostic and test software. Often, control, diagnostic, and test software requires an order of magnitude larger person-hour effort than does application software [8]. Conventional hardware software co-design methods assign a token interest in the issue of software required for control, diagnostic and test purposes, and attempt to catch all integration issues under the term "interface." The approach shown in Figure 1(ii) represents a "true" HW/SW

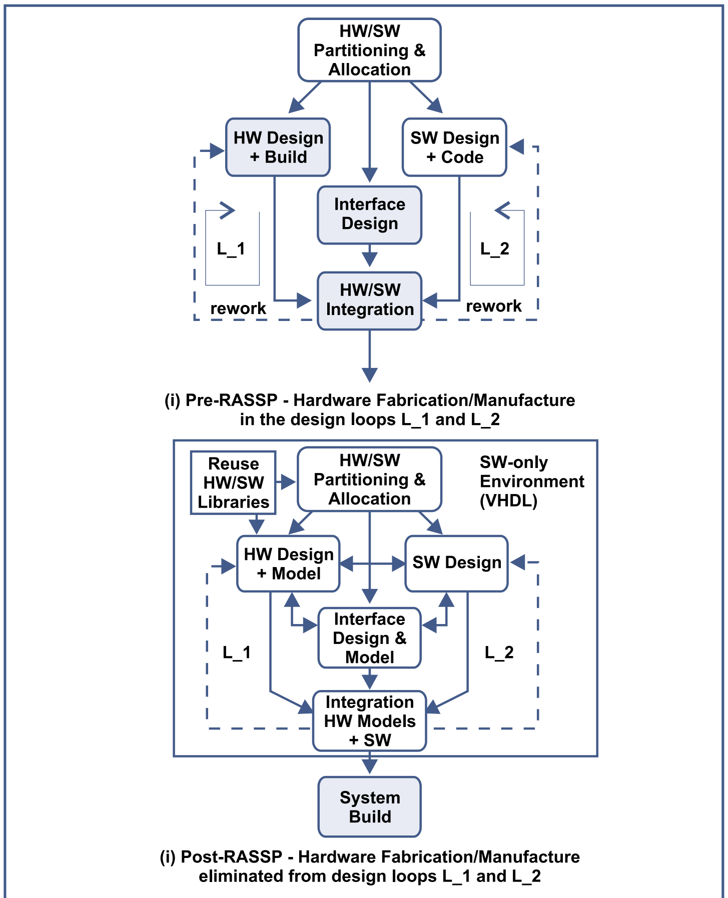

HW/SW Partitioning & Allocation
HW Design + Build
SW Design + Code
Interface Design
L_1
L_2
rework
rework
HW/SW Integration
(i) Pre-RASSP - Hardware Fabrication/Manufacture in the design loops L_1 and L_2
Reuse HW/SW Libraries
HW/SW Partitioning & Allocation
SW-only Environment (VHDL)
HW Design + Model
SW Design
Interface Design & Model
L_1
L_2
Integration HW Models + SW
System Build
(i) Post-RASSP - Hardware Fabrication/Manufacture eliminated from design loops L_1 and L_2

Figure 1. HW/SW Codesign - (i) Current practice (1993-1994) and (ii) True HW/SW codesign. Note elimination of hardware fabrication, assembly and board/system level manufacture from the design loops. Software can be tested on virtual hardware that is also concurrently being designed. Savings in time and cost, capability for customer input, and concurrent life-cycle support and upgrade planning is facilitated. Shaded areas imply hardware (board/MCM level).

[1] This research was supported by Advanced Research Projects Agency (ARPA), Department of Defense, under the RASSP program, 1994-1997.

codesign wherein software models (in a HDL such as VHDL) of HW are provided to the SW developers and the entire software is designed and tested and integrated with the HW models long before any hardware is fabricated or manufactured. Thus, the design loops L_1 and L_2 are quick, and require no hardware fabrication & engineering cost, and in addition provide capability for complete system design using a process known as *virtual prototyping* [1,5,10].

2.2 Showstoppers

The assumption, of course, is that libraries of full-behavioral HW models in SW are accurate, available and interoperable, and that simulation times can be kept manageable. VHDL can be used with advantage in this true HW/SW codesign philosophy — one that embraces a hardware-less system design. Recent experience with hardware-less HW/SW codesign has shown that it is efficient, often reducing time for HW/SW integration to a matter of weeks, and also allows rapid upgrades, together with savings in cost [9]. Once virtual prototyping is completed, it is expected that that pathway through which a field prototype can be manufactured, supported and upgraded will be straighforward.

3. Models for HW/SW Codesign

Several classes of models have been found suitable for HW/SW codesign. When emphasizing HW/SW integration two classes have been found particulary useful - ISA models and full-behavioral models. We will utilize the RASSP taxonomy [6] to define these two classes.

(1) *Instruction Set Architecture (ISA) Models* — An ISA model describes the function of the complete instruction set recognized by a given programamble processor, along with (and as operating on) externally known register set and memory/input-output space. An ISA model will execute any machine program for that processor and give exactly the same results as that processor (e.g., bit-true) as long as the initial states are the same for both simulation and the real system. Port registers, if modeled, are also bit-true. Instructions span multiple clock cycle, and ISA models need not contain any internal structural implementation information.

(2) *Full-Behavioral Models (FBMs)* — A full-behavioral (also known as full-functional model [10]) is a processor model that exhibits all documented timing and functionality of the modeled component, without specifying internal structural implementation details. Thus, the full-behavioral model is more detailed than the ISA model in that it includes clock-edge timing information in addition to functionality. A number of full-behavioral processor models are available from Georgia Tech's RASSP Techbase effort [5]. The issue of creating ISA and FBMs will be examined next.

3.1 Populating VHDL Model Libraries

While complete or incomplete gate-level VHDL models are sometimes available from vendors, and are accurate for use as ISA and FBMs, a number of limitations exist — (1) gate-level models are very slow in terms of simulation times, (2) reveal confidential component design (intellectual property) information, and (3) HW/SW codesign assumes that the hardware component is continuously being designed (e.g., changing instruction set, optimized behavior, etc), and thus the gate-level description does not exist. Thus, the focus is on creation of behavioral models of complex parts. Commercial Instruction-set simulators (ISS), which can provide debug information for processors, have limited applicability within a VHDL-based environment (without wrappers and loss in efficiency) where multiple models at varying levels of detail are co-simulated during the top-down design process. In addition, they do not allow redesign of the hardware component, a trend that is increasing finding favor in application-specific markets (e.g., use of core-based functional design of DSP ASICs [2]).

The current approach to model development is best described in [3,4,10]. All these approaches model the internal and external microarchitecture of the component behaviorally from manufactured-supplied data (or via abstraction to higher levels of functional and timing information from gate-level descriptions). This is a manual, time consuming (in person-years), and error-prone (i.e., verification) operation, and often equivalent to designing the component all over again. While we have used this approach, and continue to use this approach in developing ISAs and FBM models, an investigation into automated generation of these models was long overdue.

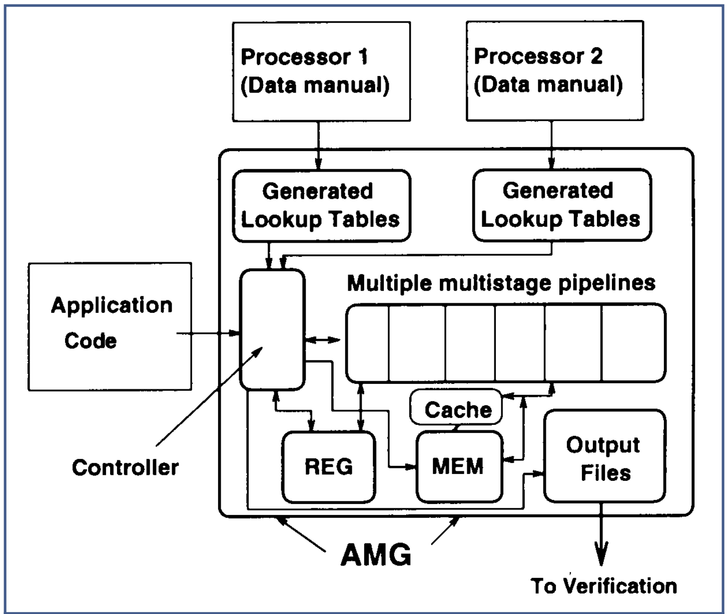


Figure 2. Automated Processor Model Generation (AMG) for Simulation

3.2 A New Approach - Autogeneration

An alternative approach to developing ISAs and FBMs that is automated is described in Figure 2. The processor being modeled (or designed) is described by parametrized generalized time-stationary [2] pipelines (single or multi-), associated memories/ registers, and a generalized controller. The user-defined or vendor-supplied information on the instruction set, architectural constraints (hazards, timing), are captured in terms of processor-specific input data files. These parametric input data files then are automatically converted to lookup tables (LUTs). The LUTs are utilized by the AMG to generate the control (timing) and functional information from the input application instruction stream. We have used this approach to synthesize behavioral models of the PowerPC 601 RISC processor and an implementation will be described in the next section.

3.3 A New Approach - Iterative Synthesis

The approach described in Figure 3 describes the process flow for automating the iterative synthesis of application-specific processors. Here the instruction-set of a programmable processor can itself be customized and iteratively designed during the HW/SW codesign process. The application drives the iterative instruction-set and architecture codesign (which are captured from input data files as LUTs) by the AMG. The controller, pipeline, and associated logic of the AMG are then simulated to measure performance on the target application. After optimization of the instruction set and timing, the AMG may be synthesized using commercial RTL-level or behavioral synthesis tools. Application-specific functional libraries can also be used with advantage when combined withVHDL and the emerging VITAL standards for sign-off quality timing simulation. Future papers will discuss and document the approach of Figure 3.

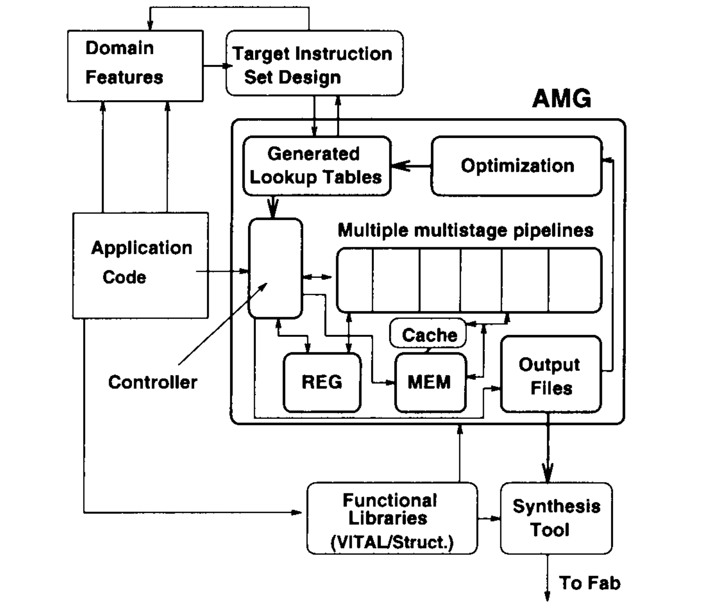


Figure 3. Automated Processor Model Generation (AMG) for Iterative Synthesis

4. Automated Model Generator - AMG

The automated model generator (AMG) is an ISA or FBM model that accepts the application instruction stream and processor-specific data in the form of input tables, that are processed internally to provide all documented functional and timing characteristics as output files. The same AMG can be reused for creating models of multiple versions of the same chip, or independent families of processors.

4.1 Structure of the AMG

The automated ISA model generator consists of six major "blocks," as described on the following page (See Figure 4).

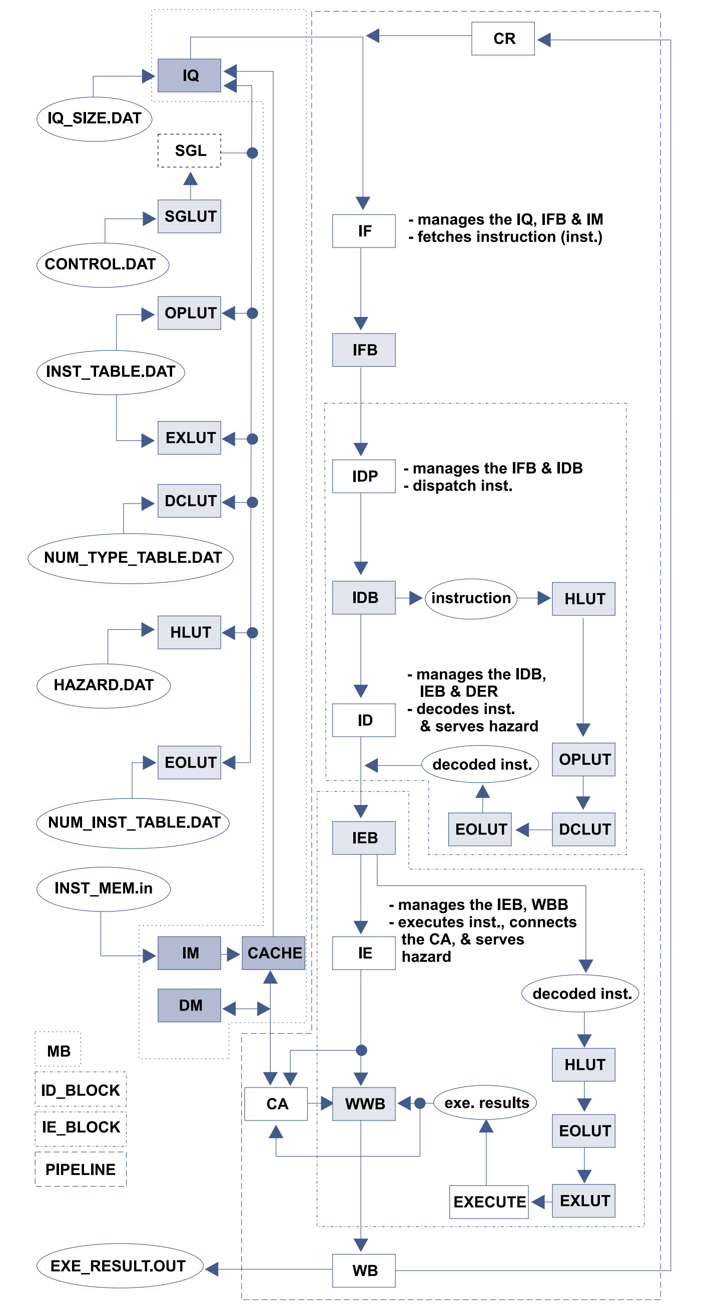


Figure 4. The Anatomy of Georgia Tech's Automated VHDL Model Generator (AMG). Hardware vendor or designer-supplied data is obtained as *.DAT, which are then automatically converted to lookup tables (LUTs) by the AMG. The output file is the behavior of the hardware executing the software (functional values and timing).

1. **Pipeline:** A single pipeline for a RISC processor consists of the following six stages — (1) Instruction Fetch (IF), (2) Instruction Dispatch (IDP), (3) Instruction Decode (ID), (4) Instruction Execute (IE), (5) Cache Access (CA), and (6) Write back (WB). These stages were implemented as procedures within a VHDL process description of the pipeline.

2. **Memory Block (MB):** The MB consists of an *instruction queue (IQ), instruction and data memories (IM & DM),* and a *cache (CACHE).*

3. **Data Register Block (DRB):** The DRB consists of a number of register arrays (DER, ECR, CWR), including a general purpose register (GPR) to allow storage for resolution of pipeline data hazards. A number of 32-by-32 bit data register arrays are also reserved for the user.

4. **Control Register Block (CRB):** The CRB consists of register arrays (CR, HIR (hazard information registers), SCR (system control registers), HDR (hazard destination registers) to control various stages of the pipeline.

5. **System Generating Logic (SGL) Block:** The SGL converts specific input data (i.e., in form of tables.dat) from manufacturer or instruction-set designer into Lookup Tables (LUTs) that can be used by the AMG. Thus, information about differing processors can be converted to a standardized internal representation that can then be utilized by the AMG in generating instructional function and timing. The six automatically generated internal LUTs are opcode lookup table (OPLUT) containing opcodes and extended opcodes for user-defined instructions, a decode lookup table (DCLUT) containing information on the bit length of the opcode and other instruction fields, an execute lookup table (EXLUT) that stores information for the execution latencies and the identification of every instruction to map into an executable location in the IE, a hazard lookup table (HLUT) containing information on data hazards of registers and memory, an extended opcode lookup table (EOLUT) consisting of data related to extended opcodes, and a system generation lookup table (SGLUT) that is used by the SGL. It may be iterated that the SGL automatically creates these LUTs based on manufacturer or designer-supplied processor or instruction-set information.

6. **Stage Buffer Block (SBB):** The SBB consists of buffers for stages of the pipeline (e.g., IFB, IDB, WBB, IWB, etc).

4.2 Operation of the AMG

We now discuss the operation of the AMG as follows —

1. **Step 1 (Fetch and Dispatch):** An instruction if fetched from the IM and brought to the IQ and CA in the pipeline. The IF fetches the instruction from the IQ and stores it in the IFB. The IDP then initiates the dispatch of the instruction from the IFB and translates it into the IDB. In order to decode this instruction, the opcode or the extended opcode is first extracted from the instruction. The instruction type is then docoded from the information available in the lookup table OPLUT.

2. **Step 2 (Decode):** The ID then obtains the extended opcode information from the EOLUT and the instruction format from the DCLUT using the decoded instruction type information as a key. In the final step at the ID, the instruction is dissassembled, the information disseminated, and valid instruction fields are stored in the IEB. After completing the decode operation, the ID checks for data dependency on the current instruction. The information stored in DER is utilized for this check, and the information is propagated to the hazard registers, HIR and HDR, with operate in conjunction with the HLUT. Operands for the operation are put in the IE buffer (IEB).

3. **Step 3 (Execute):** The IE begins operation if the IEB is nonempty. The IE updates the GPR and the DER, and picks out appropriate information from the EXLUT — i.e., instruction execution latencies, location of procedures, requirements for cache access for executing the function or process, and then executes the procedure (the AMG currently supports upto 1024 user-defined operations). The result is then stored in the WBB or sent to the CA (if cache access is needed). The HIR and HDR are then updated to allow hazard resolution for the subsequent instructions in the pipeline.

4. **Step 4 (Cache Access and Write Back):** The CA then reads/writes data from/to the cache in case of a cache hit, or the DM in case of a cache miss. The WB updates the CWR through the DER or ECR before writeback. If the instruction is processed by CA, the CWR is updated by the ECR, else, it is updated by the DER (resolving hazards between IE and CA). The result is then written to the GPR and all hazard conditions

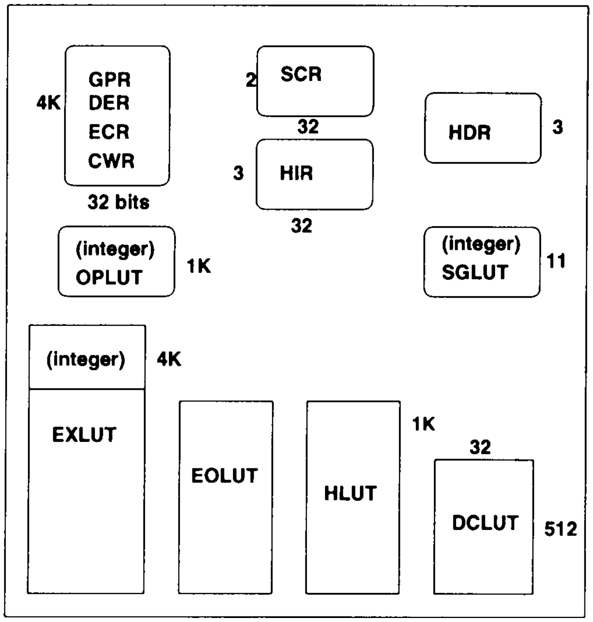


Figure 5. Register and Memory Data Structures

Description of the Instruction Timing & Test Bench

TABLE 1. Instruction Timings for the Test Bench 'A'

Addr. of Inst.	Inst.Name	rD/BI	rS/rA/rB	Instruction Cycles																													
				1	2	3	**4**	5	6	7	8	**9**	10	**11**	**12**	13	~	46	47	**48**	49	**50**	**51**	52	**53**	54	~	87	88	**89**	90	**91**	**92**
0	"add"	rD=1	0 / 2 / 3	f	d	e	w																										
1	"mulx"	rD=2	0 / 1 / 3		f	d	e	e	e	e	e	w																					
2	"ldx"	rD=3	2 / 0 / 0			f	.	.	.	.	d	e	c	w																			
3	"add"	rD=2	0 / 3 / 3								f	.	d	e	w																		
4	"divx"	rD=1	0 / 2 / 3										f	d	e	e	~	e	e	w													
5	"strx"		3 / 1 / 2											f	.	.	~	.	d	e	c	w											
6	"brx"	BI=3															~		f	d	.	e	w										
7	"addi"	rD=2	0 / 2 / 0														~			f	.	>											
3	"add"	rD=1	0 / 3 / 3																			f	d	e	w								
4	"divx"	rD=1	0 / 2 / 3														~							f	d	e	~	e	e	w			
5	"strx"		3 / 1 / 2														~								f	.	~	.	d	e	c	w	
6	"brx"	BI=3															~										~		f	d	.	e	w

"f" : Instruction fetch & dispatch stage
"d": // decode stage
"e" : // execute stage
"c": // cache access stage
"w": // writeback stage
">": purge Instruction on the pipeline
"." : stall on the pipeline

TABLE 2. Characteristic of the instructions

Instruction name	Opcode	Extended opcode	Instruction Type	Latencies
add	31	248	1	1
addi	14	0	2	1
ldx	34	0	3	1
strx	31	215	4	1
mulx	31	107	1	5
divx	31	331	1	36
brx	18	0	5	1

TABLE 3. Description of the instruction fields

Instruction Type	rD	rS	r1	r2	BI	C1	Eop	SIM	UIM	RC	AA	C2	RSV
1	5	0	5	5	0	1	9	0	0	1	0	0	0
2	5	0	5	0	0	0	0	16	0	0	0	0	0
3	5	0	5	0	0	0	0	0	16	0	0	0	0
4	0	5	5	6	0	0	9	0	0	0	0	0	1
5	0	0	0	0	24	0	0	0	0	0	1	1	0

rD : Destination Register
rS : Source Register
r1 : working Register 1
r2 : working Register 2
BI : immediate field for branch
C1 : control bit 1
Eop : extended opcode
SIM : signed immediate field
UIM : unsigned immediate field
RC : record bit
AA : absolute address bit
C2 : control bit 2
RSV : reserved bits

TEST_BENCH 'A' (INPUT)

```
8                                   :end addr.+1
                                      of inst. stream
0                                   :start addr.
                                      of inst. stream
01111100001000100001100111110000    :"add",   rD=1
01111100010000010001100011010110    :"mulx", rD=2
10001000011000100000000000001111    :"ldx",   rD=3
01111100010000110001100111110000    :"add",   rD=2
01111100001000100001101010010110    :"divx",  rD=1
01111100011000010001000110101110    :"strx",  rD=1
01001000000000000000000000001100    :"brx",   BI=3
00111000010000110001100110011110    "addi",   rD=2
```

TEST_BENCH 'A' (OUTPUT)

```
 4  :end of cycle time   "add",   rD=1
 9  :        //          "mulx" rD=2
11  :        //          "ldx",   rD=3
12  :        //          "add",   rD=2
48  :        //          "divx",  rD=1
50  :        //         "strx",   rD=1
51  :        //          "brx",   BI=3
53  :        //          "add",   rD=2
89  :        //          "divx",  rD=1
91  :        //         "strx",   rD=1
92  :        //          "brx",   BI=3
```

Figure 6. Results from the AMG-generated PowerPC 601 Model

caused by the current instructions are void. The WB also generates the *output file* with the necessary user-specified information on execution times and functional results required from the model.

4.3 Implementation of the AMG

To test the AMG we first implemented the AMG in VHDL, and successfully modelled a subset of the ISA of the PowerPC 601 with a single pipeline. More recently, the AMG has been generalized to model multiple concurrent pipelines and other processors (e.g., i860 and ADSP 21060).

In one of our PowerPC 601 variations of the AMG, that is fully operational, each memory within the MB was implemented as a 32 bit-vector array (same as the instruction length). The IQ IM, and DM are 64-by-32, 8K-by-32, and 20K-by-32 arrays, respectively. The SBB was implemented as four buffers, one of which is the IEB that is a 256-integer variable buffer for maintaining latency and executing function information in the IE stage, the others maintain bit-vector and one integer type variable for maintaining the latencies of other pipeline stages. Figure 5 summarizes the sizes of the other register and memory arrays utilized in our implementation. Note that the user of the AMG can tailor the pipeline to suit his/her implementation specifications, and can also utilize more than one pipeline within the AMG (i.e., the PowerPC 601 has three pipelines — integer, floating, and branch). The AMG currently has been implemented in about 5K lines of uncommented VHDL source code.

4.4 Performance of the AMG - PowerPC 601

Figure 6 describes the performance of a PowerPC 601 model generated by the AMG. The input source code is described in Test Bench A, and was input to the AMG. The AMG then generates the function and timing behavior via output files (shown also in Figure 6), and via VHDL *signals* (that are displayed on a VHDL simulator spreadsheet in Figure 7). Tables 1 and 2 in Figure 6 describe the detailed clock-cycle resolved operations of the pipeline for the PowerPC 601. The exact timing for the completion of each instructions are also shown. In Figure 7, for instance, the multiply is described in the *decode* buffer as 7*c*4118*d*6, and has a latency of 5 clock cycles, which are successively decremented as shown on the signal INST.EXE.CYC.1. Typical instructions executed per second on the virtual model generated by an unoptimized AMG were in the order of 500-1000 for single pipelines, and less for multiple pipelines (10-200). For a 1000 instruction test bench, the execution times on a Sparc10 workstation were; multiple pipeline AMG (242.95 sec), PowerPC with multiple pipelines (235.55 sec), Single pipeline AMG (18.0 sec), PowerPC with single pipeline (1.45 sec). The time required to generate a model is limited only by the time it required to enter the input.DAT tables from the manufacturer's data sheets (or in the case of iterative synthesis, from the designer), and took about a person month for the PowerPC. The AMG consists of about 5K lines of VHDL source code and utilized the Vantage VHDL Spreadsheet at Georgia Tech's DSP Laboratory.

5. Summary and Conclusions

Models have been shown to very useful in the system prototyping process, often reducing HW/SW design and integrations costs by a factor of four or more. The contributions of this paper are as follows -

1. A new method for automated generation of full-behavioral and ISA models for complex pipelined processors has been proposed. We believe that this is the first such proposal and its implementation.
2. A new method for iterative synthesis, where the instruction-set of a processor can be customized to the application software, utilizing true hardware/software codesign is proposed.
3. Successful demonstration of the proposed method for automated generation, using the PowerPC 601 as an example. Our results show that the speeds in instructions per second range between 500-100 for single pipelines and 5-100 for multiple pipelines and comapare well to manually generated behavioral models. The time required for model development is, however, shorter, requiring a few person-months for an ISA model (without interface timing), as opposed to 1-3 person-years for the manual method of model generation.

Further optimization of the automated model generation process is an ongoing investigation.

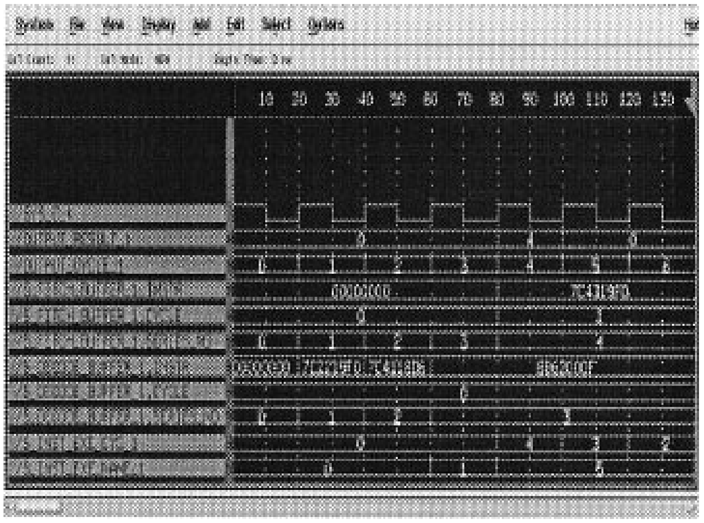


Figure 7. VHDL Simulations of the AMG-generated PowerPC 601 Model Using Testbench of Figure 6 Confirm the Behavior of the Processor

Acknowledgements

Thanks to M. Rubeiz of Wright Patterson Labs (USAF) for carefully reviewing the manuscript.

References

[1] M. Richards, “The Rapid Prototyping of Application-Specific Signal Processors Program,’’ Proc. of *First Annual RASSP Conference*, August 1994.

[2] V. K. Madisetti, *VLSI Digital Signal Processors*, IEEE Press, Piscataway, NJ, May 1995.

[3] Z. Navabi, “Using VHDL for Modeling and Design of Processing Units,” Proc. of *5th Annual IEEE ASIC Conference and Exhibit*, pp. 315-326, 1992.

[4] L. Maliniak, “Process Builds Accurate VLSI Behavioral Models,” Electronic Design, pp. 63-70, May 3, 1993.

[5] V. Madisetti, T. Egolf, S. Famorzadeh, L-R. Dung, “Virtual Prototyping of Embedded DSP Systems,” Proc. of *IEEE ICASSP 95*.

[6] C. Hein, T. Carpenter, P. Kalutkiewicz, V. Madisetti, “RASSP VHDL Modeling Terminology and Taxononomy - Revision 1.0,’’ Proc. of *Second ARPA RASSP Conference*, July 1995.

[7] C. Myers, R. Dreiling, “VHDL Modeling for Signal Processor Development,” Proc. of *IEEE ICASSP 95*.

[8] V. Madisetti, J. Corley, G. Shaw “RASSP: Current Practice (1993) E&F Model and Challenges,’’ Proc. of *ARPA Second RASSP Conference*, July 1995.

[9] The RASSP Information Server - WWW URL http://rassp.scra.org.

[10]T. Egolf, V. Madisetti, S. Famorzadeh, P. Kalutkiewicz, “Experiences with VHDL Models of COTS RISC Processors in Virtual Prototyping for Complex System Synthesis,’’ Proceedings *VHDL International Users’ Forum (VIUF), Spring 1995*.

Vijay K. Madisetti and Yong-Kyu Jung
ECE,
Georgia Tech.
Atlanta, GA 30332-0250
vkm@ee.gatech.edu

Mississippi State Develops On-Line FPGA VHDL Model Generator

Robert Reese and J. Scott Calhoun

Abstract

As part of our RASSP Tech Base VHDL modeling and distribution objectives Mississippi State University (MSU) has developed and released an on-line FPGA VHDL model generator.

The objectives of the MSU tech base contract are to provide VHDL models for COTS parts, primarily concentrating on PLDs, RAMS, and ROMS. To date, we have released VHDL models for several parts in these general families, including several standard PLD models (22V10, several PALs), standard RAM/ROM models, dual port RAMS, and some general bus interface glue logic models. We have also made available several VHDL packages which are useful for creating additional part models within these general families.

Another objective of our program was to advance beyond low-complexity PLD models and provide a VHDL model for a field programmable gate array. Recently, we made an alpha release of a VHDL model for the Xilinx X4000 FPGA family. The remainder of this article will discuss our approach to this model.

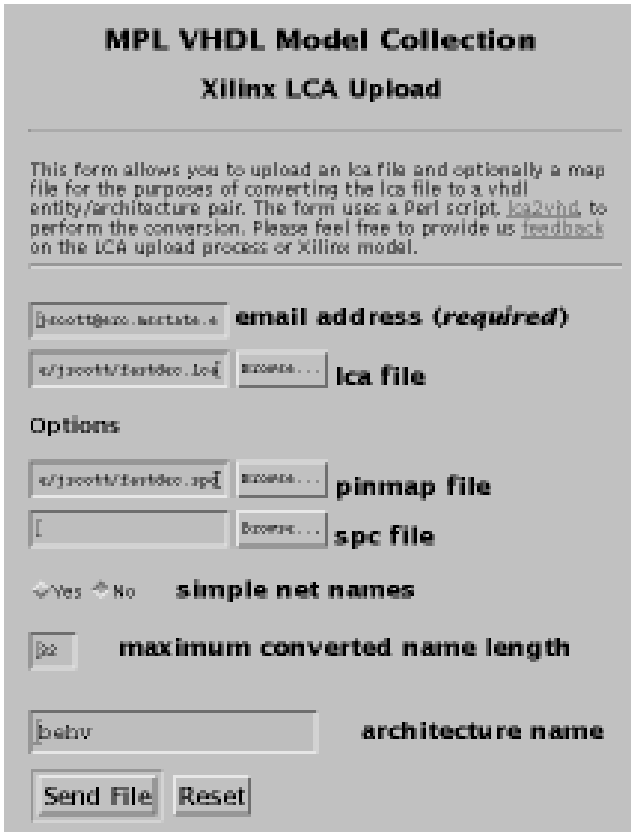
1. Xilinx X4000 FPGA

The Xilinx X4000 family is a static RAM based FPGA. The basic logic cell is called a Configurable Logic Block (CLB) and contains two 4-input lookup tables, two D flip-flops, and dedicated carry logic. The lookup tables can implement two separate 4-variable functions; the output of the tables can also be combined to form a third logic function. The lookup tables can also be used as an asynchronous SRAM, synchronous SRAM (X4000E) or dual port SRAM (X4000E). Combinational outputs can be registered via the flip-flops if desired.

I/O is handled via a versatile Input Output Block (IOB) which have several configuration options such as registered/non-registered on the input or output signals, tri-state output, programmable output slew rate, pullup/pulldown on output, and output polarity. There are several other logic resources on the chip as well - an on-chip oscillator, fast decoders, tri-state buffers, pullups, high-drive buffers, startup logic, and boundary scan capability.

2. Modeling Goals and Approach

Our goals in modeling the X4000 was to provide a VHDL model which supported logic functionality, timing, system level features, and have good performance. System level features include simulation of capabilities not normally supported during gate level development - i.e, startup emulation, boundary scan support, and in-system programming. The SRAM programming of the X4000 gives the part a dynamic reconfigurability capability via in-system programming. After looking at the problem, we decided not to support dynamic reconfigurability in our VHDL model of X4000



MPL VHDL Model Collection

Xilinx LCA Upload

This form allows you to upload an lca file and optionally a map file for the purposes of converting the lca file to a vhdl entity/architecture pair. The form uses a Perl script, lca2vhd, to perform the conversion. Please feel free to provide us feedback on the LCA upload process or Xilinx model.

jscott@erc.msstate.e **email address (*required*)**

s/jscott/fastdec.lca Browse... **lca file**

Options

s/jscott/fastdec.spc Browse... **pinmap file**

Browse... **spc file**

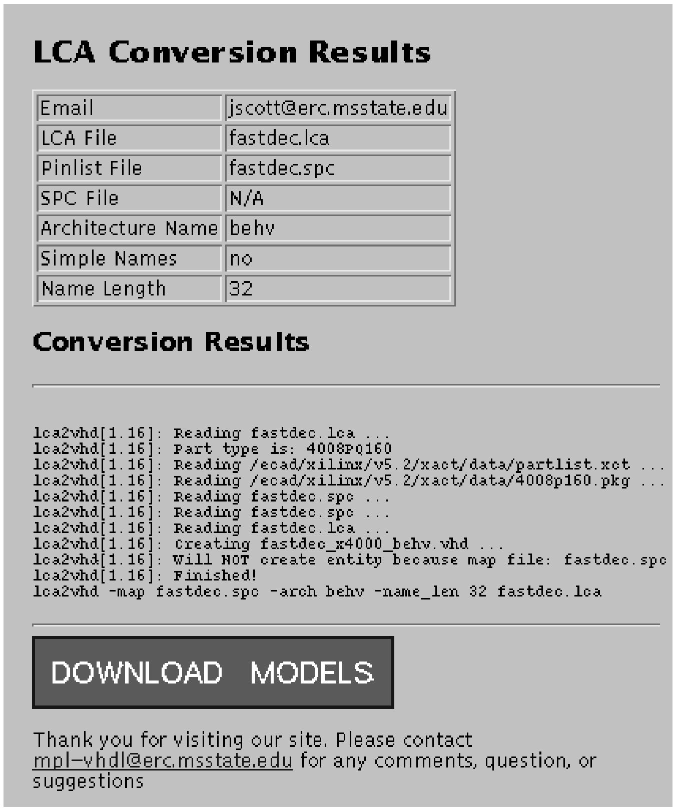
◇Yes ◇No **simple net names**

32 **maximum converted name length**

behv **architecture name**

Send File Reset

Figure 1. LCA Upload/Model Generation/Model Download Form



LCA Conversion Results

Email	jscott@erc.msstate.edu
LCA File	fastdec.lca
Pinlist File	fastdec.spc
SPC File	N/A
Architecture Name	behv
Simple Names	no
Name Length	32

Conversion Results

```
lca2vhd[1.16]: Reading fastdec.lca ...
lca2vhd[1.16]: Part type is: 4008PQ160
lca2vhd[1.16]: Reading /ecad/xilinx/v5.2/xact/data/partlist.xct ...
lca2vhd[1.16]: Reading /ecad/xilinx/v5.2/xact/data/4008p160.pkg ...
lca2vhd[1.16]: Reading fastdec.spc ...
lca2vhd[1.16]: Reading fastdec.spc ...
lca2vhd[1.16]: Reading fastdec.lca ...
lca2vhd[1.16]: Creating fastdec_x4000_behv.vhd ...
lca2vhd[1.16]: Will NOT create entity because map file: fastdec.spc
lca2vhd[1.16]: Finished!
lca2vhd -map fastdec.spc -arch behv -name_len 32 fastdec.lca
```

DOWNLOAD MODELS

Thank you for visiting our site. Please contact mpl-vhdl@erc.msstate.edu for any comments, question, or suggestions

Figure 2. On-Line Model Generation Output

for several reasons:

a. Supporting dynamic reconfigurability could severely impact model performance, both in execution time and in memory footprint. This conflicts with our goal of achieving good performance.

b. Dynamic reconfiguration is not a capability which is used that often.

c. Supporting dynamic reconfigurability would require a non-disclosure agreement with Xilinx concerning the format of the programming bit-stream which might hinder model distribution.

Not supporting dynamic reconfigurabilty meant that a static VHDL model could be generated from one of the intermediate logic representations which are created during the design/mapping process. Xilinx supports two netlist formats which describe mapped logic - XNF (Xilinx Netlist Format) and LCA (Logic Cell Array). XNF is an intermediate gate level description while the LCA netlist uses the on-chip logic primitives (CLBs, IOBs, tri-state buffers, etc). We chose the LCA format because we felt it would be easier to support system level features (startup, boundary scan) at this level and because the generated package/speed-grade timing information refers directly to these primitives.

3. Model Library and Generator

A VHDL X4000 module library was written to support the logic primitives. Currently, the first release (May 96) of this library supports all logic functionality except for the startup and boundary scan logic. Some timing functionality is included but work is still in progress in this area. A Perl5 script named 'lca2vhd' converts an LCA file into a VHDL structural model which uses the X4000 module library. Any net delay information present in the LCA file (produced by Xilinx back annotation) is represented in the VHDL model. X4000 module timing information is read from a separate file generated by the Xilinx mapping process and is based upon selected package type and speed grade. The 'lca2vhd' script converts net names in the LCA file to VHDL-compatible names; a command line option allows user control over name mapping which proves useful in generating vectored signal names. The X4000 module library has been tested with both Mentor Quick-VHDL and Vantage VHDL development environments.

4. On-line Model Generation

In keeping with the on-line release mechanism used by MSU to distribute its VHDL models, the FPGA model generator takes this paradigm to the next level. The FPGA VHDL model library for the X4000 is distribtued via the World Wide Web (WWW) download at

http://www.erc.msstate.edu/mpl/vhdl/html/models/library/xilinx.html

Once the X4000 library is downloaded and compiled, VHDL models of specific X4000 designs can be generated via the on-line model generator at

http://www.erc.msstate.edu/mpl/vhdl/html/models/library/xilinx/lcaform.html

This unique form takes advantage of an emerging HTTP feature allowing files to be uploaded to the web server {Note: this feature is currently supported by Netscape Version 2.0 or greater}. LCA files (with optional name mapping and timing files) are uploaded to the MSU RASSP webserver. The uploaded files are fed to the lca2vhd generator. The generated models are then tarred and compressed and available for download. The model generator form along with and example of the generator output execution is illustrated in Figure 1 and Figure 2.

This unique application takes a significant step towards the use of on-line CAD systems which may be prevalent in the future. We hope to collect usage data that will help the DoD and CAD industry determine the value of these on-line applications. An example of the model generator output for a CLB component used within a design is given below.

```
    CLB_PD: x4000CLB
        GENERIC MAP (
            WD_G4 => 3.6 ns,
            WD_G3 => 1.9 ns,
            WD_F4 => 1.3 ns,
            WD_F3 => 1.1 ns,
            WD_F1 => 2.1 ns,
            WD_C1 => 3.7 ns,
            WD_K => 1.4 ns,
            MGeneration  => MGeneration,
            XGeneration  => XGeneration,
            Ref  => Ref&":CLB_PD",
--  Config  F4:F4I  G2:  G3:G3I  X:  Y:  XQ:QX  YQ:
FFX:K:RESET
    FFY:RESET DX:H
--                  DY: F:F3:F4:F1
--  G:G3:G4 H:F:G:H1 H1:C1 DIN: SR: EC: RAM:
    CARRY: CIN: COUT: CDIR:
        CLBTags => CLBTagsArrayData(PD_blk),
        CLBTiming => CLBTimingRecordData,
        RAMTiming => RAMTimingRecordData,
--  F = (F1*~((F3 + F4)*~(F3*F4))), G = (G3*G4),
    H = (~H1*(G + F))
            CLBFuncs => CLBFuncsArrayData(PD_blk)
        )
        PORT MAP (
            G4 => N_1I120_PATH_NET_23, -- $1I120/PATH/
            NET_$23
            G3 => N_1N237,              -- $1N237
            G2 => open,
            G1 => open,
            F4 => N_1I120_PATH_C_1002_4,  -- $1I120/PATH/
            C_1002_4
            F3 => N_1I120_PATH_N_437, -- $1I120/PATH/
        N$437
            F2 => open,
            F1 => N_1I120_PATH_NET_38,          -- $1I120/
            PATH/NET_$38
            C1 => N_1N116_2,           -- $1N116_2
            C2 => open,
            C3 => open,
            C4 => open,
            K => N_CLK,                         -- CLK Y => open,
            YQ => open,
            X => open,
            XQ => N_1N237,              -- $1N237
            COUT => open,
            CIN => open,
            GSR => GSR_LCA2XNF
        );
```

In the above generic map for 'CLB_PD', the 'CLBFuncs' generic contains the lookup table and carry logic mapping which is

```
    PD_blk =>
        (funcF => "0000000010011001",  -- F1(MSB),F2,F3,F4
        inputs
        funcG => "0001000100010001",  -- G1(MSB),G2,G3,G4
        inputs func
        H => "00101010",          -- F(MSB), G, H1
        funcC => (OTHERS => '0')      -- carry logic function
    ),
```

The 'CLBTags' generic contains a structure defining the particular CLB configuration. Timing data is passed within the 'CLBTiming' and 'RAMTiming' generics with net delays specifed via the various 'WD_*' generics.

5. Future Work

Our current goals (in order):

a. Complete all timing functionality.

b. Add X4000E support; this variant of the X4000 adds synchronous SRAM and dual-port SRAM capability in the CLB.

c. Add startup emulation and boundary scan support.

Our schedule calls for completion of this work by September '96.

Robert Reese and J. Scott Calhoun
Microsystems Prototyping Laboratory
Engineering Research Center
P.O. Box 6176
Mississippi State, MS 39762
reese@erc.msstate.edu

A Proposed Design Objectives Document for Object-Oriented VHDL

David L. Barton and
Jean-Michel Berge

1. Introduction

The Study Group on Object Oriented VHDL is continuing its work within the Design Automation Standards Committee (DASC). It is now considering a preliminary version of a Design Objectives document. This document contains the objectives that will guide and constrain the study group in its efforts to produce a definition of Object Oriented VHDL. Before we examine this document, we will set it in context. We will then set out its various sections, and discuss them

There are three parts to the efforts of the Study Group:

1. To determine a methodology using object oriented design methods and techniques for VHDL designs.
2. To establish if there are any short-term object oriented extensions that can be presented to the VHDL Analysis and Standardization Group (VASG) for consideration in the 1998 restandardization effort.
3. To assemble a long term definition of VHDL extended with object oriented constructs.

The Design Objectives (D.O.) document will support the third of these activities. This is based on the fact that VHDL provides many good features and support efficient design methodology. However, there are certain limitations in the power of the language which can be (at least partially) addressed by Object Oriented Extensions.

This document tries to identify these different objectives and to sort them into two categories: high priority and low priority. The semantics of these categories is the following:

- High priority objectives, which must be considered first. If one of these objectives cannot be reached, the Study Group must justify the omission, for example because of significant difficulties to implement it.
- Low priority objectives will be considered second. They will be implemented only if the cost of implementation is extremely low, and any decision to implement one of them must be accompanied by a rationale.

The first part of the Design Objectives document is a glossary, which will be presented next.

2. Glossary

The Glossary is an important part of any document. In particular, it is an important part of a document that is the product of a variety of different engineers from different disciplines. The first version of the glossary simply attempts to define the basic terms of object oriented programming. These are given below.

Class — An abstraction which helps in describing a system by means of objects (instantiation of classes) which communicate by triggering methods.

Object — A specific instance, or instantiation, of a class.

Method — An operation, or procedure, that is associated with a class, and which therefore operates on the objects of that class.

3. High Priority Design Objectives

The following design objectives are proposed as high priority ones. They are given in no particular order, below. With each is given a rationale, and a brief explanation if necessary.

3.1. Add Inheritance Mechanism(s) to VHDL

Rationale: VHDL does not allow the designer to incrementally modify existing “objects’’ (i.e. entities, types, etc.) and create other ones which are only slightly different. This requires the designer to write two or more) complete pieces of code. This has implications in terms of source code testing and validation.

The key issue here is avoiding this copy & paste coding method.

Inheritance mechanisms let a current class inherit information from another class, called the mother class (and implicitly from all the mother's ancestors).

Inheritance is a way to improve reusability. It is a static mechanism, which may be resolved before simulation. Multiple inheritance (inheriting from two or more mother classes) is a separate issue, and is a low priority D.O. Inheritance may trigger:

- Inheritance of all methods of the mother class.
- Inheritance of all attributes of the mother class.
- Redefinition of an existing method of the mother class.
- Addition of a new method to the current class.
- Addition of a new attribute to the current class.

3.2. Method Call or Message Passing

Rationale: The only communication mechanism of VHDL today is based on the signal value resolution semantics. At a high level of abstraction, dealing with this communication mechanism very often implies knowledge of which protocol will be used to exchange data and coding that protocol in terms of signal assignments. This yields over-specification and restricts the scope of application of VHDL.

Communication in OO-VHDL should include a higher level of abstraction which only requires the target's object name, the method to be called and its parameters.

3.3. Add Type Polymorphism to VHDL

Rationale: Strong typing is certainly an excellent mechanism to guarantee a high level of safety of the code; however, the possibility of explicitly inhibiting (at least partially) this mechanism has to be considered for different reasons:

- While progressing from the very beginning of the design cycle (the very first “specification’’) to the final implementation (the “synthesizable’’ description), the type of data is usually refined. This leads to many problems, such as introduction of conversion functions or modification of interface types (ports or generics), that have consequences for encapsulation.
- Furthermore, a strong typing at a high level of abstraction very often implies giving more information than necessary in order to anticipate compatibility with lower levels of abstraction. This leads to over-specification.
- Finally, strong typing sometimes leads to less reusability of code by requiring development of the same algorithms for different (but often closely related) types.

It might be interesting to verify at compilation time type compatibility; for example a 6 bit integer may be extended to an 8 bit integer.

As a remark, we can see two different ways of considering polymorphism:

- Dynamic polymorphism is implemented by message passing. Given the name (or handle) of an object, another object can send it a message without knowing the type of the target object. For example, an object can send a “increment’’ message to a counter without knowing if it's a DCD counter or a 6 bit counter.
- Static, relative to type, to soften in some cases the hard-typing of VHDL. Hindley-Milner type systems, as used in MHDL, are examples of practical static polymorphic systems.

For 90% of cases, only a type compatibility check and an easy to do static conversion at compilation time is required to assure type compatibility.

3.4. Keep VHDL Concurrence

Rationale : One of the strongest aspect of VHDL is its ability to manage both the sequential and the concurrent worlds. Objects should keep this quality (i.e. : objects shouldn't be considered as single thread). Sequential and concurrent code should be freely combinable.

3.5. Add Class Libraries to VHDL

Rationale: One of the main benefits of Object Oriented Techniques in the software domain is the high degree of reusability that libraries of objects can provide. Five kinds of “design units’’ are the basic elements of current VHDL libraries, and none of them can be considered as a “pure object" or a “pure class’’ (entity/architecture are not flexible enough and packages cannot be instantiated). OO Extensions must extend this somehow.

3.6. Easiness of Use, Methodology

OO-VHDL should fit the hardware view of the system and be usable by Hw/Sw designers.

One of the most important aspects of the object oriented development is a strong need for a complete methodology. This methodology has to handle all intermediate steps of a design cycle using object oriented concepts. This includes steps where object descriptions (and simulations) are mixed with (existing?) VHDL descriptions (and simulations).

4. Low Priority Design Objectives

These items are less developed than the high priority objectives. Further development will come during the Study Group deliberations. There is no particular order to this list.

4.1. Add Broadcasting to VHDL

One of the main characteristic of object-level communication is that you only need to know the name of an object and the path itself as in VHDL signal communication. Broadcasting is a way to communicate to a set of other objects without knowing their names.

4.2. Add Dynamic Creation/Disappearance of "Objects"

An object should not necessarily exist from the beginning to the end of the simulation.

4.3. Multi-Inheritance

This is the faculty to inherit from more than one mother class. Multi-inheritance implies to define how to solve inheritance conflict (for example: inheriting the same attribute twice).

4.4. Add an Exception Mechanism

Exceptions are a control mechanism existing in other languages. They are very often used to interrupt a regular treatment to execute specific error treatment. A general “reset’’ section may be an example of the use of exception mechanism in hardware.

4.5. Better Documentation Capabilities

Several participants have expressed a desire for better documentation capabilities in VHDL. This would aid class re-use, as well as help make the written classes self-documenting.

5. Additional Remarks

In addition to the explicit Design Objectives, the Study Group has also discussed whether OO-VHDL should provide some kind of linkage to further development steps of some sort. This would include at least synthesis; it is synthesis that is mentioned most often. This may be comprised of “designing a synthesizable VHDL source from OO-VHDL source.’’ It is not clear how this will be managed; however, it may be connected with method management.

6. Conclusion

We need to emphasize the preliminary nature of this design objectives document. At the time of writing, it has been neither reviewed nor approved by the Study Group. Nevertheless, it is put forward as a strawman for use by the Working Group in its deliberations. We believe it accurately reflects the current thinking of the Study Group in general.

There may be movement of design objectives between the low priority and high priority categories, and additional objectives may be added. The Study Group will review this document at the meetings in conjunction with the 1996 Design Automation Conference, and will continue to refine the document in the coming months. Eventually, it will form the basis of language change proposals and of the OO-VHDL definition.

David L. Barton
Intermetrics, Inc.
7918 Jones Branch Dr.
McLean, VA 22102
dlb@severn.wash.inmet.com

Jean-Michel Berge
CNET-France Telecom
CNS/CIT
Chemin du Vieux Chene
38240 Meylan France

The EDA Standards Roadmap and the EDA Industry Council

John Teets

Introduction

In 1996 EDAC, SEMATECH, and CFI with ARPA funding support jointly sponsored a workshop to develop an industry-wide Roadmap for development of standards within Electronic Design Automation. The initial workshop was held March 20-21, 1995 at the Westin Hotel in Santa Clara, California. As described later in this document, the 1996 version of the EDA Standards Roadmap is now available in printed hardcopy form, as well as online on the web.

Increased chip densities and the design of higher performance systems have outgrown the current era of loose collections of EDA tools. Designs across the next decade will demand integrated and highly interoperable systems that allow the designer to traverse back and forth among sophisticated CAD tools at all levels of design from architecture to implementation. Keeping pace with this advancement and the required productivity improvements will necessitate cooperative work across the industry, toward creating and adopting standards to meet EDA system needs in the areas of designer productivity and design complexity management.

EDA Roadmap "Top Ten Projects" Recommended

Technology Transfer:

1. Chip Data Representation (CHDStd)
2. Synthesizable Subsets (RTL Subsets)
3. Delay Project (DCS/DCL)
4. Software Licensing Policy
5. Open Modeling Forum (OMF)

1997 Roadmap Development:

1. PCB/MCM (Above Chip)
2. Test Standards
3. System Level Design and Verification
4. Design Reuse
5. Design System Environment
6. Design Complexity Management

1. Goals

The goal of this effort was to identify and formally document an EDA Standards Roadmap as a staged sequence of development, which must occur over the next decade to meet the requirements of Design and Test of both semiconductors and electronic systems. This Roadmap considers the target goals as well as a flexible co-existence and migration strategy to the goals from where industry is today.

2. Scope

The scope of this Roadmap covers EDA Design and Test Requirements in areas of productivity and complexity management as identified by the "SIA 1194 National Technology Roadmap for Semiconductors" (NTRS) and the SRC Semiconductor White Paper Report "Design Needs for the 21st Century: White Paper," Sept., 1994. The context is on standards and modeling which cut across all technology areas relating to electronic design.

The Roadmap provides focus on productivity and complexity management requirements and dose not address any specific

Electronics Companies:

Joseph Borel, SGS Thomson Microelectronics
John Darringer, IBM
William Evans, Lucent Technologies
Jan-Olof Kismalm, Ericsson
Lance Mills, Hewlett-Packard
L. J. Reed, Motorola

EDA Vendor Companies:

Joe Costello, Cadence
Aart deGeus, Synopsys
Alain Hanover, ViewLogic
Wally Rhines, Mentor Graphics

Semiconductor Companies:

Lambert van den Hoven, Phillips Semiconductor
Greg Ledenbach, SEMATECH
Robert Rozeboom, Texas Instruments (Chairman)
Gadi Singer, Intel
Kinya Tabuchi, Mitsubishi Electric
Hitoshi Yoshizawa, NEC

Standards Groups/Government:

Andrew Graham, CFI
Randy Harr, ARPA

The EDA Industry Council Members

CAD tool functional algorithm requirements. Further, while the sponsors are predominately U.S.-based, the Roadmap is not restricted to U.S. company inputs. It addresses worldwide standards requirements in the following areas:

- EDA CAD System Integration and Interoperability - Design and Data Management
- Technology Libraries and Models

The purpose of the Roadmap is to set direction and priority on industrial and government investments into these requirement areas, specifically recommending redundant competitive efforts that should be converged, and cooperative efforts for which investments in multiple developments serve best to achieve needed goals. Furthermore, the Roadmap provides direction to industry to enable proprietary EDA develop-ment and a base on which CAD groups can plan their proprietary CAD system integrations based upon open EDA standards.

3. Roadmap Organization

Development of the Roadmap was performed under the direction of CFI, EDAC, and SEMATECH by an invited group of technical experts (Roadmap Development Working Groups) from a cross-section of industry, and the results were approved in October 1995 by a select body of key industry leaders (Industry Council) who have the authority and influence to assure implementation of the result.

4. The EDA Industry Council

The EDA Industry Council first met in August 1995 for the purpose of reviewing and adopting the work of the EDA Standards Roadmap effort.

The mission of the Industry Council, as presented by Robert Rozeboom, chairman of the Council, is outlined below:

- Mission: (why does the Council exist)

Promote the adoption and use of open EDA practices and technology in electronic hardware design.

- Vision: (description of the desired future state)

Sufficient data reuse and tool interoperability exists in the EDA industry such that users can focus their energy on satisfying customer needs with new technologies.

- Strategy: (how will this vision be accomplished)

Identify and define directions for critical EDA Standards needed to accomplish business objectives. Facilitate the implementation and promote the use of these standards by aligning industry resources.

- Tactics: (specific tasks)

Establish and maintain the EDA Standards Roadmap based on the NTRS Roadmap and user requirements.

Identify and use "fast track" standards processes for implementation and adoption of standards

For more details on the above, those readers with access to the web are referred to the EDA Industry Council Home Page at http://www.cfi.org/ic, and the online version of the Standards Roadmap which is accessible from there. The Roadmap can also be downloaded in PostScript form from the CFI ftp site via anonymous login to ftp.cfi.org to /public/Cfi/Development/Roadmap/EII/Roadmap.ps. A presentation to the Industry Council which overview the "top ten" projects is also available for download from the IC Home Page.

Your feedback on the Roadmap and EDA Industry Council activities is encouraged and welcome! Please contact John Teets (teets@cfi.org) for additional information.

John J. Teets, II
CFI
4030 W. Braker Lane, Suite 550
Austin, TX 78759
teets@cfi.org

A Formal Model of Digital Systems Compatible with VHDL

Philip A. Wilsey, Sheetanshu L. Pandey and Kothanda Umamageswaran

Abstract

Hardware Description Languages (HDLs) play an important role in the design and verification of digital devices. Unfortunately most HDLs are informally defined and thus, they are frequently defined only partially, with some aspects of the language subject to varying interpretation. The formal models project is developing formal models and theories for the standard hardware description language VHDL. Included in the project are a collection of theories for defining (i) when a VHDL description is well-typed, (ii) equivalent static structures within the language standard, (iii) the semantics of VHDL's evaluation (or simulation time) behaviors, and (iv) semantic preserving rewriting operators that are used to optimize the analysis and parallel simulation of VHDL.

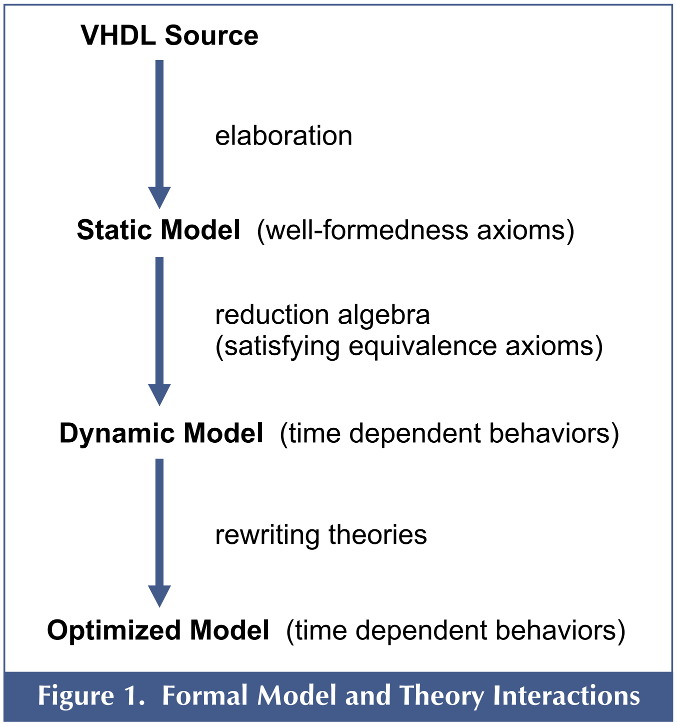

VHDL Source
elaboration
Static Model (well-formedness axioms)
reduction algebra
(satisfying equivalence axioms)
Dynamic Model (time dependent behaviors)
rewriting theories
Optimized Model (time dependent behaviors)

Figure 1. Formal Model and Theory Interactions

1. Project Overview

Besides a formal syntax definition, few formal semantic models for Hardware Description Languages (HDLs) are ever constructed. Furthermore, informal (English) specifications of "so called" standard languages allow for numerous interpretations where only one is desired. This paper reports our efforts to construct formal models for the hardware description language VHDL. In particular, the project develops the following models and theories (Figure 1):

1. *Static Model*: The static model is a set theoretic definition of post-elaborated VHDL models. Included in the static model are axioms that define when a VHDL model is well-typed (*e.g.*, the data type of the target and expression in a signal assignment statement match).
2. *Equivalence Axioms*: The equivalence axioms formally define equivalent structures in VHDL. The structures of VHDL included in the equivalence axioms are only those that are formally described to be equivalent in the language standard. For example, the language standard defines the equivalent process statement representation for a concurrent signal assignment statement.
3. *Reduction Algebra*: While equivalence axioms define equivalent structures, the reduction algebra defines operators that transform between equivalent VHDL structures. The operators of the reduction algebra thus implement transforms that satisfy the constraints of the equivalence axioms. A mechanized theorem prover has been used to show that the rewriting operators correctly adhere to the constraints of the equivalence axioms.
4. *Dynamic Model*: The dynamic model is a reduced form, set theoretic model derived from the static model. The dynamic model is derived from the static model using operators from the reduction algebra. Using the PVS theorem prover, this reduction has been shown to be complete and idempotent. Included in the dynamic model is a formal, declarative specification of the dynamic behaviors described by a VHDL model (hereafter called the *dynamic semantics*). The dynamic semantics is formed using an interval temporal logic and defines only the (signal and variable) state space defined by a specific VHDL model.
5. *Rewriting Operators*: The rewriting operators are a collection of transforms that we use to optimize VHDL for parallel simulation. The transforms are shown to preserve the dynamic semantics of VHDL and are embedded in a VHDL analyzer/optimizer/code generator for optimizing the performance of the QUEST project time warp simulator. Initially, we have developed 16 rewriting operators to merge VHDL process statements and early experiments have shown speedups as high as 2.2.

In the remainder of this paper, we briefly describe our technology transfer activities (Section 2) and highlight the significance of the static model technology developed as part of this project (Section 3). Additional details on other aspects of this project can be found in the papers enumerated in the bibliography at the end of this paper. The project is also summarized on the web at the following URL: *http://www.ece.uc.edu/~paw/rassp*.

2. Technology Transfer

The formal modeling project includes efforts to support the transfer of technology into the practicing community of VHDL-based design and analysis of digital systems. These activities include the application of the formal models to optimize parallel simulation (described above in the rewriting operators) and the insertion of

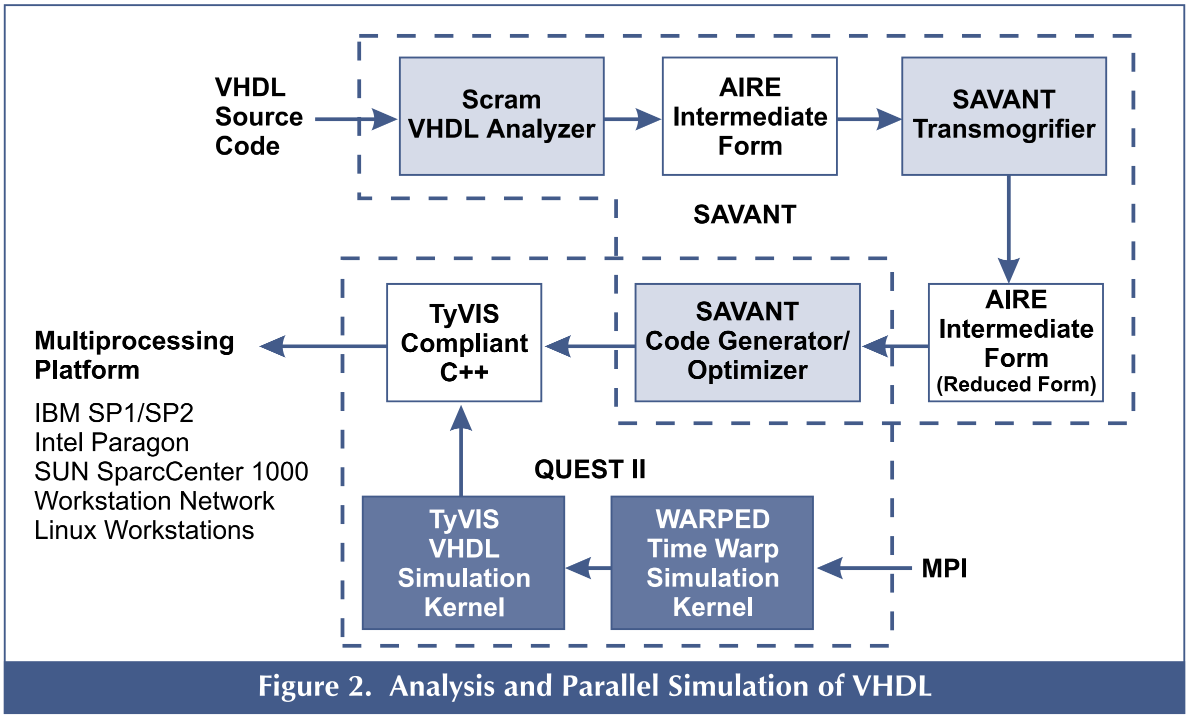


Figure 2. Analysis and Parallel Simulation of VHDL

the technology into other VHDL analysis activities at the University of Cincinnati (Figure 2). In particular:

- The static model, well-formedness axioms, and reduction algebra are being used to aid VHDL analysis. The well-formedness axioms are being used in the type-checker of the SAVANT VHDL analyzer; the static model and reduction algebra is being used to define a reduced form intermediate representation (AIRE) that is being jointly developed with John Willis. The AIRE project is planning to develop and standardize the VHDL intermediate form. A VHDL intermediate file format is also planned.
- The dynamic model is being used to define the TyVIS parallel VHDL simulation kernel. The rewriting operators are being used by the VHDL optimizer/code generator in the SAVANT/ QUEST projects.

3. Impact of the Static Model and Its Reduction

Many of the transforms that are formally developed in the static model exist in an informal form in the VHDL language reference manual. Some exist solely as additional reductions that can be used to simplify and optimize CAD tool construction. For example a simulation code generator can be greatly streamlined by the use of these transforms. More precisely, assume that a front end analyzer has been constructed to parse VHDL into the static model form and further, that a set of reduction procedures have been implemented to rewrite the model form according to the reduction algebra. By reducing the VHDL into its core constituent elements, a simulation code generator need only recognize the, much smaller, core language elements (the entire set of concurrent statements can be eliminated from the code generator's vocabulary).

The reduction algebra also simplifies the construction of a dynamic semantics for VHDL. In fact, several of the operators in the reduction algebra were developed specifically to simplify the construction of the dynamic model. For example, rewriting signal assignment statements so that the right-hand-side (rhs) is simply an expression instead of a waveform is a direct consequence of a desire to simplify construction of the dynamic semantics.

In addition, in the construction of the dynamic semantic model, we have also experienced unexpected benefits from the static model and its transformations. In particular, the representation of transport delays in signal assignment statements as inertial delay with a reject pulse limit of 0 nanoseconds has allowed us to derive a new approach for marking transaction lists. In particular, we have been able to develop a marking scheme that relies on constraints on time intervals rather than on a sequential algorithm that update transaction lists.

Finally, we expect to see additional benefits in the optimization of CAD tools as the formal models develop. For example, the static model and its transforms might well impact the design space explored by a synthesis tool. As another example, the new marking scheme described in the previous paragraph may significantly impact the performance of parallel simulation subsystems. While we expect to see additional benefits, further developments of the semantic models and the associated transforms must be developed and applied in the construction and optimization of CAD tools. We are exploring these avenues in several other research projects.

Bibliography

[1] S.L. Pandey, D.M. Benz, and P.A. Wilsey, “Formalizing the Static Structures of HDLs for the Optimization of CAD Tools,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*. (submitted).

[2] S. Pandey, K. Umamageswaran, and P.A. Wilsey, “A Complete Reduction Algebra for VHDL,” *International Conference on Formal Methods in Computer-Aided Design* (FMCAD), November 1996. (submitted).

[3] S. L. Pandey, K. R. Subramanian, and P. A. Wilsey, "A Semantic Model of VHDL for Validating Rewriting Algebras," *EuroMicro '96*, September 1996. (forthcoming).

[4] P. A. Wilsey, D. M. Benz, and S. L. Pandey, "A Model of VHDL for the Analysis, Transformation, and Optimization of Digital System Designs," *Conference on Hardware Description Languages* (CHDL'95), 611-616, August 1995.

[5] T. McBrayer and P. A. Wilsey, "Process Combination to Increase Event Granularity in Parallel Logic Simulation," *9th International Parallel Processing Symposium*, 572-578, April 1995.

[6] P. A. Wilsey, "Formal Models of Digital Systems Compatible with VHDL," Working Document, Computer Architecture Design Laboratory, University of Cincinnati, 1994 (revised 1995, 1996). (available on the www at http://www.ece.uc.edu/~paw/rassp).

This research was supported in part by the Defense Advanced Research Projects Agency and monitored by the Air Force Wright Laboratory under contract number F33615-93-C-1315.

Philip A. Wilsey, Sheetanshu L. Pandey
and Kothanda Umamageswaran
Computer Architecture Design Laboratory
Dept. of ECECS, PO Box 210030
University of Cincinnati
Cincinnati, OH 45221-0030
paw@ece.uc.edu

Calendar of Events

Digital Design 2000: A Workshop on Innovations in System Design and Impact on Academic Curriculum For more information: vkm@ee.gatech.edu	**October 4, 1996**	**Lockheed Martin ATL Labs** **Camden, NJ**
ICSPAT - DSP World Expo For more information: dsp@mfi.com	**October 7-10, 1996**	**Boston, MA**
Applied Imagery Pattern Recognition (AIRP) Workshop '96 For more information: schaefer@gmu.edu	**October 16-18, 1996**	**Washington, DC**
1996 International Test Conference "Test and Design Validity" For more information: itccon@aol.com	**October 20-24, 1996**	**Sheraton Washington Hotel** **Washington, DC**
4th Annual IEEE Workshop on Real-Time Applications For more information: salinas@starbase.nl.nuwc.navy.mil	**October 21-25, 1996**	**Montreal, Canada**
VHDL International Users' Forum (VIUF) Fall 1996 Conference & Exposition For more information: erol@ee.duke.edu	**October 27-30, 1996**	**OMNI Hotel** **Durham, NC**
IEEE Workshop on VLSI Signal Processing For more information: http://www.hpl.hp.com/imaging/vlsi96/	**Oct 30 - Nov 1, 1996**	**San Francisco, CA**
DARPA/VI VHDL Educator's Workshop For more information: info@rassp.scra.org http://rassp.scra.org/VHDL.WORKSHOP/	**November 10-12, 1996**	**San Jose, CA**
1996 International Conference on Computer-Aided Design (ICCAD) For more information: icpubpap@dac.com	**November 10-14, 1996**	**San Jose, CA**

RASSP Steering Committee

DARPA (ETO)	
• **Randy Harr**	**Program Manager**
ARMY	
• **Randy Reitmeyer**	**Administrative COTR, Lockheed Martin - Advanced Technology Laboratories**
• **Arne Bard**	**Technical COTR, Lockheed Martin - Advanced Technology Laboratories**
NAVY	
• **Ingham Mack (ONR)**	
• **Gerry Borsuk (NRL)**	
• **J. P. Letellier (NRL)**	**Administrative & Technical COTR, Lockheed Martin - Sanders**
AIR FORCE	
• **Stan Wagner**	**Educator Facilitator and Technology Base**
• **John Hines**	**COTR**

RASSP Digest-Rapid Prototyping of Application Specific Signal Processors

The RASSP Digest is published quarterly and provides information for and about the RASSP Program and rapid systems development. For more information, contact Dr. Anthony Gadient or Dr. Vijay Madisetti, Editors, at the addresses below:

Anthony J. Gadient
Phone: 803-760-4082
FAX: 803-760-3349
Email: gadient@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

Vijay K. Madisetti
Phone: 404-894-4696
FAX: 404-894-4641
Email: vkm@ee.gatech.edu
Georgia Tech
School of Elec. & Computer Eng.
Atlanta, GA 30332-0250

Bryan R. Bryant
Managing Editor
Phone 803-760-3363
Email: bryant@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

For additional RASSP information including an electronic version of this publication, visit the RASSP World Wide Web site at

http://rassp.scra.org

Recognized by IEEE as one of the top three WWW sites on DSP.

IEEE Spectrum, January, 1996

RASSP E&F

SCRA • GT • UVA • Raytheon
UCinc • EIT • ADL

SCRA
5300 International Blvd.
N. Charleston, SC 29418

June
1997

Vol.4, June, 1997

RASSP Digest

RASSP - Rapid Prototyping of Application Specific Signal Processors

Focus on RASSP Educational Activities

In This Issue

Editor's Corner

Anthony J Gadient and Vijay K. Madisetti

In the past, the commercial Electronic Design Automation (EDA) and the academic/industrial research communities have been aware of the requirement for an intensive effort to study the digital system design process in its entirety; however, resource needs, fuzzy objectives, and short-time horizon have handicapped progress. Currently, the Rapid Prototyping of Application Specific Signal Processors (RASSP) program is overcoming these handicaps and is developing a number of new technologies that will lead to shorter prototyping times, improved product quality, and reduced life cycle costs.

Successfully transferring the technology being developed by the RASSP program to industry and academia is a critical component of the overall RASSP effort. To accomplish this goal, a novel, ground breaking RASSP Education & Facilitation (RASSP E&F) program was explicitly funded and a team tasked with leading the RASSP efforts to transfer technology from the RASSP program to the university and industrial communities.

To successfully transfer RASSP technology, the RASSP E&F effort must teach engineers and scientists how to use the RASSP top-down design concepts and give managers an appreciation for the potential payoff of RASSP technology, thereby creating both a technology push and technology pull. To accomplish this goal, the RASSP E&F team has adopted a multifaceted approach. This approach is designed to help push and pull individuals and organizations through the five step technology transfer process illustrated in Figure 1.

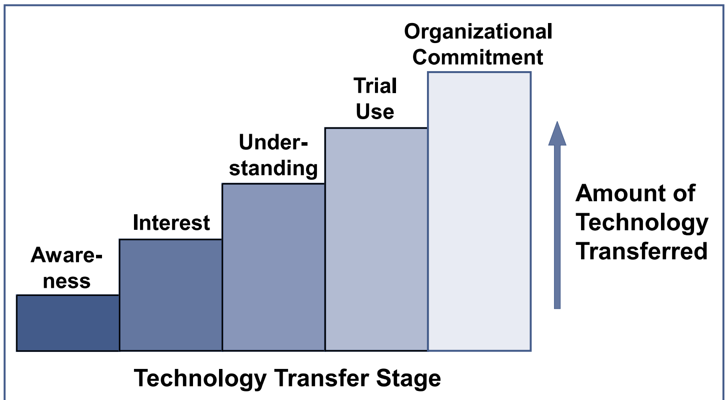


Figure 1. Technology Transfer Process

Accomplishing this process involves: (1) making available the information/knowledge necessary to progress from one stage of the process to the next, and (2) providing an education system that will assist in the transfer process. The RASSP E&F objectives in these areas are described below:

Information: develop awareness and interest in RASSP technology by providing easy access to useful and well organized RASSP-related information.

Education: educate senior management as to the potential benefits obtainable through the application of RASSP technology thereby stimulating the use of RASSP technology and the demand for RASSP trained professionals; work with universities to effect a paradigm shift in the graduate, undergraduate, and continuing professional digital system design curricula of small- and medium-sized universities to ensure that graduating students can meet industry's need for RASSP trained engineers, scientists, and managers.

The activities of the RASSP E&F program in each of these areas is detailed in this issue of the *RASSP Digest*. The newsletter starts with two invited papers that highlight the need for an activity like RASSP E&F. Professor Pettus, Chairman of the Department of Electrical and Computer Engineering at the University of South Carolina, presents an overview of the problems plaguing the university community today. Mahendra Jain, Director of VHDL International, presents industry's need for engineers educated in state-of-the-art system design techniques. These papers motivate the need to change the educational system. The article by Madisetti and Gadient entitled "A Technical Rationale for RASSP Educational Activities" explores the need to change the existing embedded digital system design education paradigm in more detail and presents the technical rationale underlying the educational efforts being undertaken by the RASSP E&F program. The article by Salinas, et. al., entitled "RASSP Educational Activities" details the RASSP E&F activities designed to enable and support the needed shift in the university education paradigm. The result is a clarion call to stimulate academic participation in a progressive educational program that adopts the latest instructional methods and industrial strength tools to revolutionize the way system design is taught in United State's colleges and universities. The article by Scharf and Karns entitled "Executive Education: Key to Implementing RASSP" details the RASSP E&F activities designed to present the business case underlying the RASSP technology. The article by Stinson, et. al., entitled "RASSP Informational Activities" details the RASSP E&F activities designed to disseminate information about RASSP.

Anthony J. Gadient
SCRA
5300 International Blvd.
N. Charleston, SC 29418
gadient@scra.org

Vijay K. Madisetti
ECE,
Georgia Tech.
Atlanta, GA 30332-0250
vkm@ee.gatech.edu

Top-Down Design of Embedded Digital Systems Educators' Workshop

August 11-14, 1997

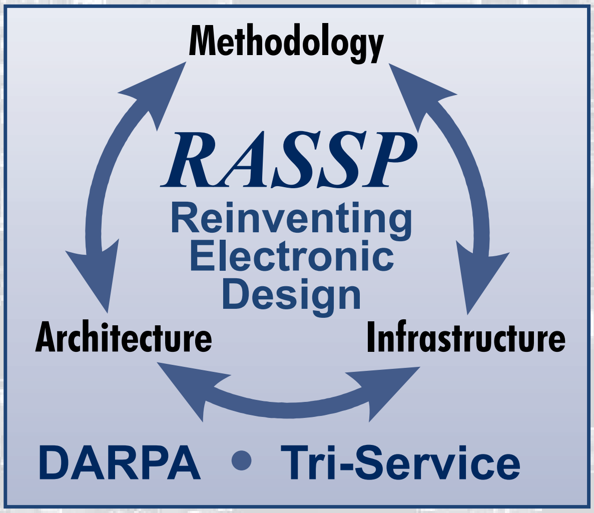
University of Virginia
Charlottesville, VA

For More Information
Registration Contact:
Eileen Johnson
(803) 760-3376
info@rassp.scra.org
http://rassp.scra.org/

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • ADL

Topics:

RASSP Methodology Overview
Executable Specifications
Cost Modeling
Performance Modeling
Virtual Prototyping
Codesign
Synthesis



Sponsored by

DARPA Defense Advanced Research Projects Agency

RASSP E&F Program

Engineering Education: Doing Business as a Business in the 90's

Robert Pettus, Professor and Chair, Electrical and Computer Engineering, University of South Carolina

1. The Inevitability of Change in Engineering Education

When I was asked to write this article, I was asked to talk about the need for changes in engineering education, given *(i)* limited resources and *(ii)* rapidly changing technology. Since we have dramatic reductions in our resources, a new Dean of Engineering, an entirely different assessment scheme from the South Carolina Commission on Higher Education, and new and totally changed Accreditation Board for Engineering and Technology (ABET) accreditation criteria on the way, I can be very passionate about this topic. However, a reasonable view is that change is an inevitable part of the engineering profession, and therefore, of engineering education. In addition to the forces which shape education in general, we are also affected by technological change and by the nature of the job market. We differ from science, for instance, in that we are seeking to solve the problems of the future as opposed to discovering the secrets of the past. For example, two of the four courses that are required in the sophomore year of the current University of South Carolina (USC) electrical and computer engineering curricula did not exist when I was a student in the early 1960's. The fundamental nature of the job market has changed over the past decade. Employers are more selective about hiring decisions. Students are more likely to have to sell their skills and experience than their degree.

On some occasions, the forces which affect the engineering profession and engineering education are aligned. For instance, in the late 1960's and early 70's, the simultaneous declines of the space program, the defense industry, and the commercial aerospace industry created a significant (negative) impact. These situations create a "burning platform" environment in which the need for change is obvious and the resistance is minimized. To a great extent, the current forces created by limited resources and rapidly changing technology have created a similar need for significant change.

2. Technological Change

Technology affects us by both the rate of the change and by its nature. Most branches of science show an exponential growth rate of 4 to 8% per year. For example, *Chemical Abstracts* took 37 years to publish its first million abstracts, 18 years to publish the second million, and 1.75 years to publish the third million[1] . Those individuals and institutions that ignore this exponential growth tend to make bad decisions, such as that made by the former head of a major (but now much smaller) computer company, who stated that he did not believe there would ever be a place for a computer in the home. Some changes, particularly new technology such as the transistor, result in changes of curricula. Other changes, in enabling technologies such as the computer, affect the manner in which we do business. The most profound current change is being created by improvements in our ability to communicate. Our educational system has been based on the concept of a central repository of information since its inception. Cellular phones, television, the internet, and other new telecommunication technologies are changing this concept. These changes have been occurring sufficiently gradually that we have not yet seen any widespread changes. However, the idea of moving the knowledge to the student as opposed to moving the student to the knowledge will likely have some profound changes over the next decade. One concept which might not make the change is the semester. When education is delivered directly to the customer, we may have to adapt to the customer's schedule. Educational institutions may have to act more like the power company in this regard.

3. Limited Resources

Engineering education has been subjected to the same down-sizing as has industry. Funding for higher education in South Carolina has been either flat or decreasing for almost 10 years now. To cite an example, the number of faculty in the USC ECE department has dropped from 23 in 1989 to 15 at the end of the Fall semester 1996. The state-appropriated operating budget has dropped to essentially $0. Limited resources, while probably more due to temporal than technological change, have been the burning platform for engineering education in the 90's.

4. Strategic Directions for Engineering Education

A number of American institutions have transformed themselves in the latter part of the 20th century. Foreign competition has forced business to become more efficient and more customer focused. This competition, particularly from the Pacific Rim, was sufficiently intense to create a tense situation. Those companies that have done the best job in changing to a new situation, i.e., reducing their costs while providing value to their customers, are also the most profitable. Engineering education (and education in general) must learn from the lessons of industry. In particular, we must become more customer focused. We must also respond positively to the changing social needs and environment of our country. In this area, the US Army provides a positive example. Like many businesses, the Army has been re-engineering. In this case, the crucible was the Vietnam conflict. The current Army is an inclusive organization which reasonably reflects the demographics of the country and which is largely free of many current societal problems, such as drug use. I believe that the Army provides some good lessons in dealing with social problems, and is a better model than industry, because it cannot use exclusion to any great extent to create quality. If education, especially public education, is to serve an appropriate role, then we must provide our services to a broad segment of the public.

About four years ago, electrical and computer engineering at USC, together with the rest of the University, began a series of changes designed to re-engineer the way we did business. The first thing we did was to rethink our basic philosophy. Our industrial advisory board had a significant impact on us during this process. The results were that we determined that we should:

1) Be good stewards of the resources under control of the department and use these resources for relevant and attainable goals.

2) Develop an awareness of the identity of our customers and take their needs into account in all decisions.

3) Promote equal opportunity and fairness in all activities.

4) Seek to add value in all relationships.

We became more of a business and began to make the appropriate changes. We reduced our number of research areas from seven to three in order to focus on our strengths. Likewise, we dropped the number of hours in the curriculum from 137 to 124 and focused on the quality of the courses as opposed to the number. Listening to our customers also led us to put more effort into non-technical skills. We now have a Writing Center, with a full time director, staffed by graduate students from the English Department's Composition and Rhetoric program. Students spend a substantial amount of time working in organized groups to learn team skills. The funds required to run these programs come from revenues generated by our research program, which has prospered when run like a business. In short, we have found that, as the title says, we must do business like a business in the 90's.

Reference

[1] "Electronics and the Dim Future of the University", Eli M. Noam, *Science*, vol 270, 13 October 1995.

Robert Pettus
Professor and Chair, Electrical and Computer Engineering
University of South Carolina
Columbia, SC 29208
pettus@ece.sc.edu

VHDL International's University Program

Mahendra Jain, Executive Director, VHDL International

VHDL International is a nonprofit organization whose mission is to cooperatively and proactively promote VHDL as a standard worldwide language for the design and description of electronic systems. With VHDL International's commitment to supporting electronic systems designers, semiconductor suppliers, EDA companies and others in the industry, VI sought to assess whether the upcoming generation of electrical engineers have tools and the training they will need to take part in the post-digital electronics industry.

In 1994, VHDL International completed a major education survey. The objective was to assess the level of VHDL knowledge of BSEE graduates and what steps VI can take to increase the VHDL knowledge in engineering schools. This survey was completed by Texas Instruments in the US. Toshiba in cooperation with Electronic Industries Association of Japan completed a similar survey in Japan.

VI found undergraduate BSEE students receive significant training in UNIX/C programming environment, but little training in specific HDLs, such as VHDL or other HDLs. Based on the results of the survey it was determined that only 14% of the US graduating seniors and 2% of the Japanese graduating seniors had a working knowledge of VHDL.

VI made a commitment to increase the level of VHDL knowledge both in the US and Japan. As part of this commitment VI and its member companies have put together a program targeted at the universities in US and Japan. As part of this university program Compass, Ikos, Mentor Graphics, and Viewlogic have donated or agreed to provide VHDL software and training materials to the universities in US and Japan. VI is also cooperating with DARPA to cosponsor VHDL workshops for university instructors so that there are enough qualified VHDL instructors to teach VHDL courses in the universities.

We have already completed the first and second phases of this university program in the US and Japan. In the first phase we sent out the education report, a contact list of VI member companies participating in the university program, a list of VHDL text books and articles, and several issues of *VHDL Times* which is published by VHDL International. This information packet was sent to about 90 US universities and 56 have responded to take part in the second phase of this program. In Japan 29 universities are participating in this program. In the second phase of this program VI has mailed VHDL learning kits with VHDL simulation software from Compass. Viewlogic has also sent their software out to participating universities in the US and Japan. Mentor Graphics and other VI member companies will be doing the same. Any university interested in VI university program should contact VI at 408-492-9806 or email at viadmin@vhdl.org.

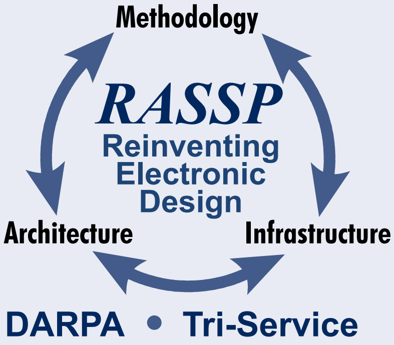
Mahendra Jain
Executive Director, VHDL International
3140 De La Cruz Boulevard
Santa Clara, CA 95054
jainm@vhdl.org

VHDL Performance Modeling Workshop

November 16th-17th
Charlottesville, VA

Purpose:

This workshop will provide an introduction to the use of VHDL in system level performance modeling. The course will include: definitions, objectives, metrics, and pervious performance modeling techniques such as Petri Nets and Queuing models of performance modeling. In addition, techniques that have been developed to build performance models in VHDL will be discussed along with the tools that are available to help automate this process. Examples of VHDL performance models will also be used to illustrate VHDL based performance modeling techniques and tools as well as show the benefits of this process. Finally, the technique of mixed level modeling in VHDL, where a portion of a VHDL performance model is refined to the behavioral or gate level and co-simulated with the remaining performance model, will be discussed.



RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • ADL

Audience:

This workshop is intended for design engineers and educators who are involved in or teach the process of system design and architectural selection at a high level and who are interested in including performance modeling of system level designs in their design flow. Attendees should be familiar with the VHDL language and VHDL modeling at the behavioral and gate levels.

Date:

November 16th and 17th, 1997

Place:

Center for Semicustom Integrated Systems Department of Electrical Engineering, Thornton Hall
University of Virginia
Charlottesville, VA 22901

Technical Contact:

Dr. Bob Klenke, Principal Scientist
(804) 924-6079
e-mail: klenke@Virginia.EDU

Administrative Contact:

Eileen Johnson, SCRA
(803) 760-3376
email: johnson@scra.org

A Technical Rationale for RASSP Educational Activities

Vijay K. Madisetti and Anthony J. Gadient

1. Introduction

This article describes, the technical rationale behind the RASSP Education & Facilitation program. In this ground-breaking effort, the Department of Defense's Advanced Research Projects Agency (DARPA) has explicitly funded technology transfer from its Rapid Prototyping of Application-Specific Signal Processors (RASSP) program to the university and industrial communities. Rather than follow the traditional passive approaches (e.g., licenses, papers, patents) for technology transfer from DoD programs to the industry and the universities, it was felt that an *active* contracted effort would meet the objectives of RASSP better and in a more timely manner.

The RASSP E&F goals may be summarized as follows:

1. Propose a relevant curriculum in system-level design, and create a high quality base of educational material based on RASSP program results to support it.
2. Propose and implement a model for technology transfer commensurate with the needs of industry and academia.
3. Utilize modern technologies, such as distributed collaboration and WWW, to provide the necessary infrastructure to support technology transfer.

The remainder of this article will discuss Bloom's Learning Taxonomy and the RASSP E&F team's Educational Maturity Model.

2. Bloom's Learning Taxonomy

Science is generally based on experimental methods that allow the formulation of general theoretical constructs. Applied sciences focus scientific theory to purposeful activity. Technology and engineering, on the other hand, put applied science to work efficiently in a process context. While science seeks basic understanding, technology and engineering are primarily goal-oriented activities in response to societal needs [3,4].

Technical and engineering knowledge can take three forms. *Descriptive* knowledge describes things as they are, usually rules, general concepts, and principles in a narrative manner. *Prescriptive* knowledge is the technical know-how gained from repeated application of descriptive knowledge, and can be captured and transferred via case studies and demonstrations. Finally, *tacit* knowledge is implicit. This encompasses "tricks of the trade", including protected and competitively sensitive knowledge. Shop floor and "skunk works" type innovations are difficult to capture, and tacit knowledge can only be learned by doing. Thus "hands on" or *proximal* learning methodologies are most suitable for transferring tacit technological knowledge.

We found Bloom's taxonomy [2] very useful in the development of a novel Educational Maturity Model (EMM) that has been used as a framework for developing the RASSP educational material to support the transfer of technical knowledge. Bloom classified learning in the classroom into the following levels.

- *Knowledge*: Student learns terminology, facts, and definitions, including benefits of applying the technology under study.
- *Comprehension*: Student can make use of ideas and material without seeing their full implication. Extrapolation to new situations is possible in limited context.
- *Application*: Student can apply knowledge to practical cases through the use of tools.
- *Analysis*: Student can break down the components of a system, and can identify hierarchies and relationships between elements. Organizational structures and assumptions (unstated) can be recognized.

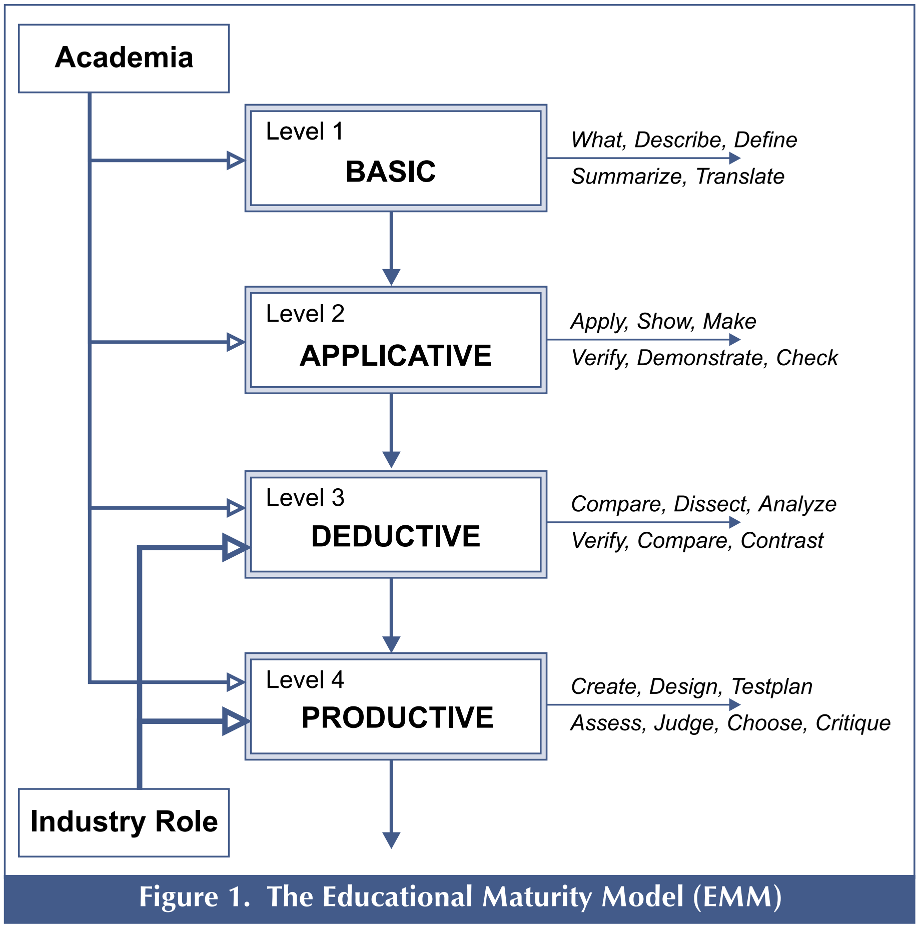


Figure 1. The Educational Maturity Model (EMM)

- *Synthesis*: Student is able to synthesize a system from start, using decomposition methods or otherwise. This include ability to produce a plan to design and implement the system, and a mechanism to verify that the plan works and will achieve objectives.
- *Evaluation*: Student can evaluate, compare, critique, and judge various alternative solutions and improve upon the product.

3. Educational Maturity Model (EMM)

Derived from Bloom's taxonomy, we developed our Educational Maturity Model (EMM) [5], Figure 1, that allows us to classify the levels of maturity of educational material (see also Figure 2).

1) *Basic* - This level of material supports knowledge and comprehension abilities on the part of the student.

2) *Applicative* - This level indicates that educational material facilitates usage of tools and application of knowledge to practical problems in limited context. Knowledge is primarily narrative.

3) *Deductive* - Supports learning of analytical aspects of technology, and the capability to apply general principles to specific cases. Prescriptive aspects of the knowledge are transferred at this level.

4) Productive - This level supports synthesis-related and evaluative aspects of learning and is the most advanced level. Included are the tacit aspects of the technology being transferred.

The underlying ideas that motivate the EMM, indicate that Level 1 (basic) can be supported by typical classroom instruction and presentation, Level 2 (applicative) can be supported by hands-on laboratories that make use of point tools (e.g., a VHDL simulator) to perform simple example problems, Level 3 (deductive) can be supported by advanced hands-on labs and notes describing the design of an advanced subsystem(s), and the most advanced level, Level 4 (productive) can be supported by material that allows the hands-on design and prototyping of actual complex systems through the use of tools and through evaluation of various trade-offs. Level 4 educational material prepares the student, with little additional on-site training, for an immediate role as a productive engineer in industry or government. Often a particular industry may hire engineers educated to Level 3 and provide on-site courses to raise the level of knowledge to Level 4. Level 4 does not stand alone but requires "Level 3 understanding" in a number of related areas of specialization, as it deals with aspects of the complete system.

The Educational Maturity Model (EMM) allows organizations to develop and evaluate training material at each of the levels. Currently, very little is done in the typical university classroom beyond Levels 1 and 2. Levels 3 and 4 are primarily outcomes of knowledge gained in industry, and would greatly benefit the quality of education in the engineering area were it included in the university curricula. Cooperative industrial training, where the student spends summers in industry, is often an attempt to substitute for Levels 3 and 4.

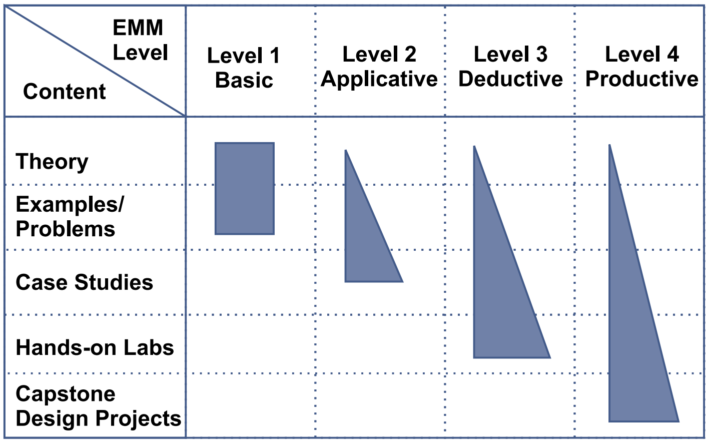


Figure 2. Relating EMM to the content and focus of educational material. Width of triangles indicate relative strengths of the content and level.

In our efforts as part of the RASSP program, in addition to Levels 1 and 2, we have attempted to ensure that the material produced would support education at Levels 3 and 4. To accomplish this, the RASSP E&F team has developed a novel module-based framework. Similar to the knowledge unit concept proposed by the Joint Curriculum Task force [1], modules are developed on specific topics and then used in the development of a new course or for updating an existing course. The attractiveness of this approach is that it is easy to insert new material into an existing course or change the emphasis of a course through the use of modules. Likewise, it is easy to develop a new course that is customized towards a specific set of goals by grouping together a collection of modules. These capabilities are extremely useful in overcoming the traditional difficulty that instructors have in inserting new courses into an existing curricula.

A typical module, illustrated in Figure 3, consists of three components. The first component is the fundamental theory underlying the topic being covered. For example, in the module on Test Technology, the theory includes a discussion of the test problem, test generation and fault simulation theory, and design for testability techniques. The second component consists of examples, problems, and case studies. This component provides simple examples that illustrate the theory and provides problems that can be used for homework exercises. The third component of a module is a hands-on laboratory exercise. The laboratory exercise is intended to rigorously demonstrate the concepts taught in the other sections of the module by providing an opportunity to apply those theories on a significant problem in a learn by doing fashion. The article by Maximo Salinas, et. al., entitled "RASSP Educational Activities" presents the RASSP E&F educational activities in more detail including a description of the RASSP E&F team's development of a comprehensive curriculum for digital system design.

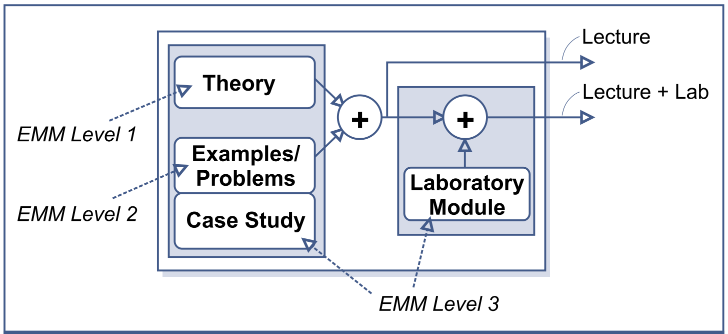


Figure 3. Module Organization

Thus, the Educational Maturity Model (EMM) allows a synergistic effort in both the creation, testing, and archiving of educational material relating to new technology developments. We have recently proposed the creation of a national digital design archive that would utilize the state-of-the-art techniques to address the issues of quality, peer review, and comprehensiveness of a library-based educational system. Course modules, simulation tools, and interactive laboratories, will now undergo a systematic classification and review process before being incorporated into the proposed National Digital Design Archive[5].

References

[1] ACM/IEEE-CS Joint Curriculum Task Force, *Computing Curricula 1991*, ACM Baltimore, MD., Order No. 201880, 1991.

[2] Bloom, B. S. (1956). Taxonomy of educational objectives, *Handbook 1: Cognitive domain.* New York: Longmans Green.

[3] Frey, R.E. (1989). A philosophical framework for understanding technology. *Journal of Industrial Teacher Education*, 27(1), 23-35.

[4] Lewis, T. and Gagel, C. (1992). Technological literacy {A critical analysis. *Journal of Curriculum Studies*, 24 (2), 117-138.

[5] Madisetti, V., Gadient, A., Stinson, J., et. al., (1997) DARPA's digital system design curriculum and peer-reviewed educational infrastructure, *Proceedings of the American Society for Engineering Education*, June 1997

Vijay K. Madisetti
ECE,
Georgia Tech.
Atlanta, GA 30332-0250
vkm@ee.gatech.edu

Anthony J. Gadient
SCRA
5300 International Blvd.
N. Charleston, SC 29418
gadient@scra.org

RASSP Educational Activities

Maximo Salinas, James Aylor, Robert Klenke, Harold Carter, Vijay Madisetti, and Anthony Gadient

1. Introduction

Today, digital system design education is focused on a limited subset of embedded digital system applications. The academic focus tends to be on applications that are limited in complexity, lack real-time constraints, and can generally be satisfied by "hardware-only" implementations due to their limited flexibility [1,3]. The design of larger systems is taught via extrapolation of this approach.

Industry needs engineers that are trained in the latest, most effective embedded digital system design technologies. To meet this industrial need, the educational modus operandi must be updated to incorporate the revolutionary new design techniques being developed in the RASSP program and elsewhere.

To ensure the successful transfer of RASSP program technologies in the longer term, RASSP technologies need to be reflected in the curricula of our academic institutions. To accomplish this goal, a novel RASSP Education & Facilitation (RASSP E&F) program was defined and a team was tasked with leading the RASSP education efforts. The RASSP E&F program was awarded in June 1994 to a team led by SCRA with team members from the Georgia Institute of Technology, the University of Virginia, Raytheon, the University of Cincinnati, Arthur D. Little and Enterprise Integration Technologies (EIT). The mission statement for the education team was fairly comprehensive:

- Educate senior management as to the potential benefits obtainable through the application of RASSP technology thereby stimulating the use of RASSP technology and the demand for RASSP trained professionals;
- Work with universities to effect a paradigm shift in the graduate, undergraduate, and continuing professional digital system design curricula of small- and medium-sized universities to ensure that graduating students can meet industry's need for RASSP trained engineers, scientists, and managers.

The education program of the RASSP E&F effort described in this article has been designed to allow the incorporation of not only the RASSP-developed technologies, but also all future technologies necessary for improved products and processes. The key strengths of this program are two-fold. First, the program is modular so that academic programs can use only those portions that are appropriate. Second, the program supports the continuous upgrade of the material, thereby making the educational material always up-to-date, and supporting the involvement of all participants in the education of system design. The latter attribute creates the potential for activities that can outlive the RASSP program.

2. Module Concept in Education

The RASSP E&F team has developed a novel module-based framework that can be used to efficiently insert the technology being generated on the RASSP program into university curricula. Similar to the knowledge unit concept proposed by the Joint Curriculum Task force [2], modules are developed on specific topics and used to develop courses. The attractiveness of this approach is that it is easy to insert new material (through the use of a module) into an existing course. This technique is extremely important because of the inertia of curricula development.

Each module consists of a comprehensive discussion of one technical sub-area, (e.g., virtual prototyping) and presents the technical details, examples, and case studies needed to obtain a thorough understanding of the topic. The specific material found in a module includes such items as presentation materials, notes on presentation materials, predefined laboratories, and homework problems with solutions.

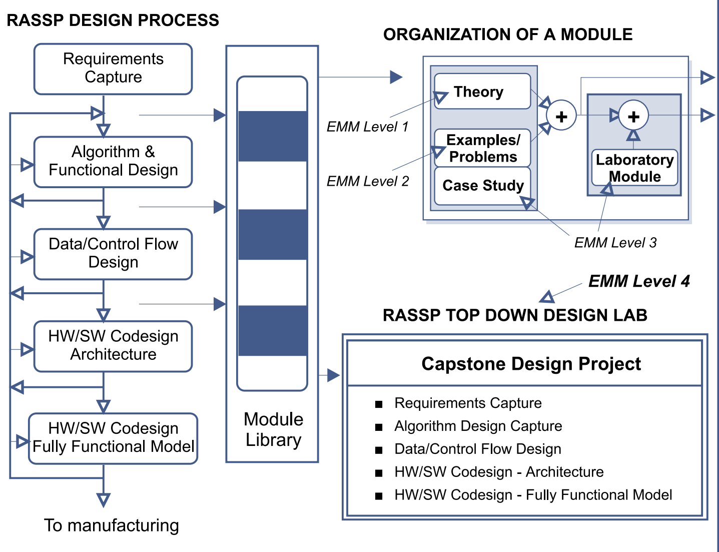


Figure 1. Module Development Process and Organization

A typical module, illustrated in Figure 1, consists of three components. The first component is the fundamental core of design principles (e.g., in a high speed IIR digital filter design module, it could outline the use on the basic operations such as retiming, pipelining, and unfolding of the flow graph). The second component consists of examples, metrics, case studies and problems. This component provides simple examples that illustrate the theory covered in the fundamental core and provides problems that can be used for homework exercises. The third component of a module is a hands-on laboratory exercise. The laboratory exercise is intended to rigorously demonstrate the concepts taught in the other sections of the module by providing an opportunity to apply those theories on significant problems in a "learn by doing" fashion.

Each module represents a unit of a course that is independent of other modules in the course (aside from prerequisite requirements).

A typical module is designed to provide three hours of lecture time. The laboratory portion of a module may actually not exist for all modules or may span multiple modules. In the ideal situation, the lab components represent a continuous, semester long design project broken into smaller pieces.

There are many advantages to encapsulating a focused amount of material in a modular fashion. These include:

- Modules can be used in a "mix and match" scenario, depending upon the particular area of digital system design as well as the target audience needs;
- As technology advances in an area, only modifications to applicable modules are necessary. This approach reduces the cost of upkeep and makes it easier to keep pace with the rapid pace of technological change;
- Modules can be easily incorporated into existing graduate or advanced undergraduate courses within a university.

The following is a comprehensive list and overview of the modules that have been developed by the RASSP E&F team. Each module is developed by the organization with the greatest strength in that technical topic. For example, the *DSP Architectures* module was developed by the Georgia Institute of Technology, whereas the *Design for Manufacturing* module is being developed by Raytheon and Arthur D. Little. Because the development of each module may be done by a different author, a standard template was developed to maintain a consistent format amongst the modules. These modules are available to instructors for use in their curricula via <***http://rassp.scra.org/***>. Note that new modules currently under development are shown in italics and are scheduled for completion in Summer 1997.

- **VHDL Basics:** an introduction to the VHSIC Hardware Description Language (VHDL), IEEE Std 1076-1993, and its fundamental concepts.
- **Structural VHDL:** a description of the use of VHDL in describing models in terms of component instantiations and interconnections.
- **Behavioral VHDL:** a description of VHDL features that can be used to describe the outputs of a component in response to changes in its inputs.
- **Advanced Constructs in VHDL:** a description of the constructs in VHDL that are more reminiscent of high-level programming languages such as file I/O, abstract data types, shared variables, etc.
- **System Level Modeling:** an introduction to techniques used for modeling systems at a high level (CPU, Memories, Interconnect, etc.) in a top-down design process.
- **Hardware/Software Codesign:** an introduction to the concepts of codesign (concurrent design) of hardware and software, from specifications, for embedded systems.
- **Hardware/Software Partitioning:** an introduction to the techniques used, in the design of embedded systems, to determine which functions are to be implemented in software on COTS processors and which are to be implemented in hardware (ASICs) and the trade-offs associated with such a partitioning.
- **DSP Architectures:** a description of various computation, communication, I/O, software, test, and maintenance architectures for embedded digital signal processors.
- **Scheduling & Assignment for DSP:** methods for allocation, scheduling and assignment of a set of software tasks in a DSP application to a selected hardware architecture.
- **DSP Algorithm Design:** a description, including examples, of a number of simulation-based functional and timing design and verification environments for design of digital signal processing algorithms.
- **Communication Protocols:** a presentation of selected communications protocols for DSP architectures geared towards understanding the relationship between them and overall system performance.
- **RASSP Methodology Overview:** an introduction to the RASSP program including a comparison of pre-RASSP and current RASSP design methodologies.
- **Virtual Prototyping for DSP Architectures:** a description of virtual prototyping (simulation based design) as applied to the design of DSP systems. Included are executable specifications, algorithm development, architecture selection, detailed design and implementation and test.
- **Virtual Prototyping using VHDL:** A discussion on how a virtual prototyping based top-down design flow is realized in VHDL. A complex design example is presented showing detailed integration and test.
- **Hardware Synthesis Overview:** an introduction on the concepts of hardware synthesis including definitions, how synthesis tools function, and general coding styles for successful hardware synthesis.
- **Libraries: Generation, Maintenance, and Reuse Overview:** an overview of problems that inhibit hardware/software reuse practice and current solutions for them. A survey of reuse metrics and a tool that tracks them is also presented.
- **Test Technology Overview:** a presentation of the fundamentals of digital systems testing including fault modeling, test generation and fault simulation algorithms, and design for testability and built-in self test techniques.
- *Requirements and Specifications Modeling:* a description of how executable specifications are derived from customer requirements, and a description of how they drive the top-down design process through regression testing.

- *Performance Modeling using VHDL:* a presentation of the environments that exist for doing simulation based performance modeling using VHDL. A discussion of hybrid modeling – the simulation of mixed performance and behavioral models – is included.
- *Enterprise Integration:* a presentation on the supporting EDA infrastructure, tool/configuration management rationale, workflow methodologies, and distributed collaboration and design environments, utilized in a top-down system-level design process.
- *Cost Analysis for Design:* a discussion of how quantitative and empirical cost models for design, implementation, test maintenance, and production can be utilized in the front-end design of embedded digital systems, in a concurrent engineering approach.
- *Robust Design for Quality:* a presentation on how products can be designed for 6-sigma quality. Included are state-of-art discussions on Taguchi methods, Monte Carlo methods, surface response, and fuzzy set methods for improving quality of products by improving their tolerance to process parameter variations with case studies.
- *Project Management:* this presentation covers the scheduling, administrative, workflow, financial, and customer-support related issues in managing large electronics system design projects.
- *Design for Manufacturing:* a description of how products and processes are designed for ease of manufacture.
- *Implementation Technologies:* a description of the various technologies available for implementation of digital systems including trade-offs and changes in the design process for them. Included are FPGAs, ASICs, Custom ICs, and MCMs**.**

After initial development, each module goes through an extensive review process beginning with an internal RASSP E&F peer review, which aligns the modules' focus for coherent course integration. Because authors have different styles, care must be taken to maintain consistency in module formats, terminology, and content, in terms of depth and detail. External review by independent experts from academia and industry helps to assure correctness and relevancy of the information captured in the modules.

The presentation material is being developed using Microsoft PowerPoint(TM), version 4.0. The slides use bulletized text, illustrations, etc. to communicate the subject matter. Associated with each slide is a "notes page." These notes provide the instructor with in-depth information including background, context, and references for further topic exploration. If desired, copies of the slides may be provided to the students. Although the hard copies tend to be in black and white, the presentations are in full color, when shown with a projection system.

3. University Education

To create the technology push for RASSP, an innovative program targeted primarily at small- and medium-sized universities has been established. As is illustrated in Figure 2, the education and research programs within universities are ultimately driven by the application needs of industry and government. These application areas determine the types of *resources* (e.g., qualified personnel) and *technologies* (e.g., research) that industry and government need from the university community. To ensure the successful transfer of RASSP technologies over the long-term, these technologies must be reflected in the curricula of the academic institutions. This approach will assure that industry's need for "resources" who understand the RASSP technology and for improvements to that technology will be met by the university community. This approach will also provide the RASSP technology push that will complement the technology pull being created through Executive Seminars. Furthermore, given the increasing rate of technological change in the area of embedded digital systems, it is important that mechanisms be established that will help assure that the academic community can meet the changing needs of industry and government.

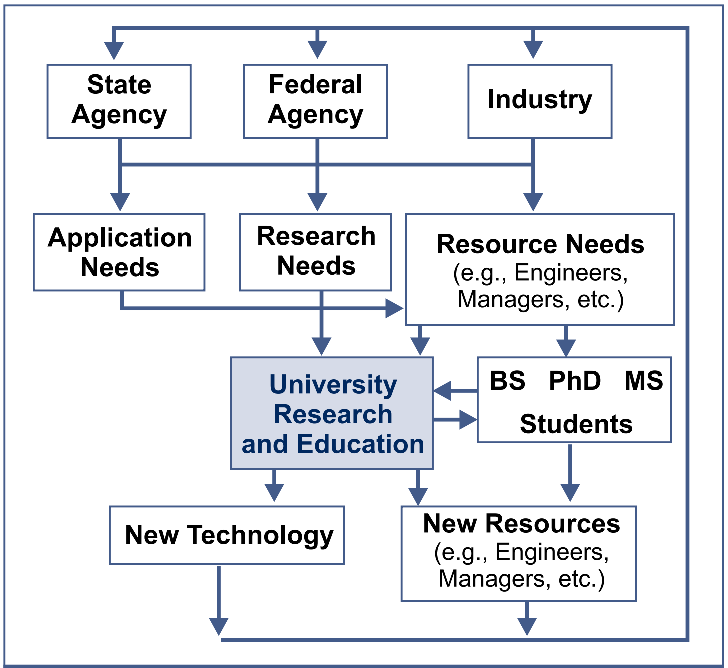

State Agency
Federal Agency
Industry
Application Needs
Research Needs
Resource Needs (e.g., Engineers, Managers, etc.)
University Research and Education
BS PhD MS Students
New Technology
New Resources (e.g., Engineers, Managers, etc.)

Figure 2. Relating Industry and Government to the University Education and Research Program

The need to update the existing curricula is evidenced by a recent survey sponsored by Texas Instruments and Toshiba. The results of this survey, designed to assess the status of hardware description language (HDL) education in engineering schools within universities in the United States and Japan [5], indicated an alarmingly low percentage of graduating seniors possessing a working knowledge of either VHDL or Verilog. Given the

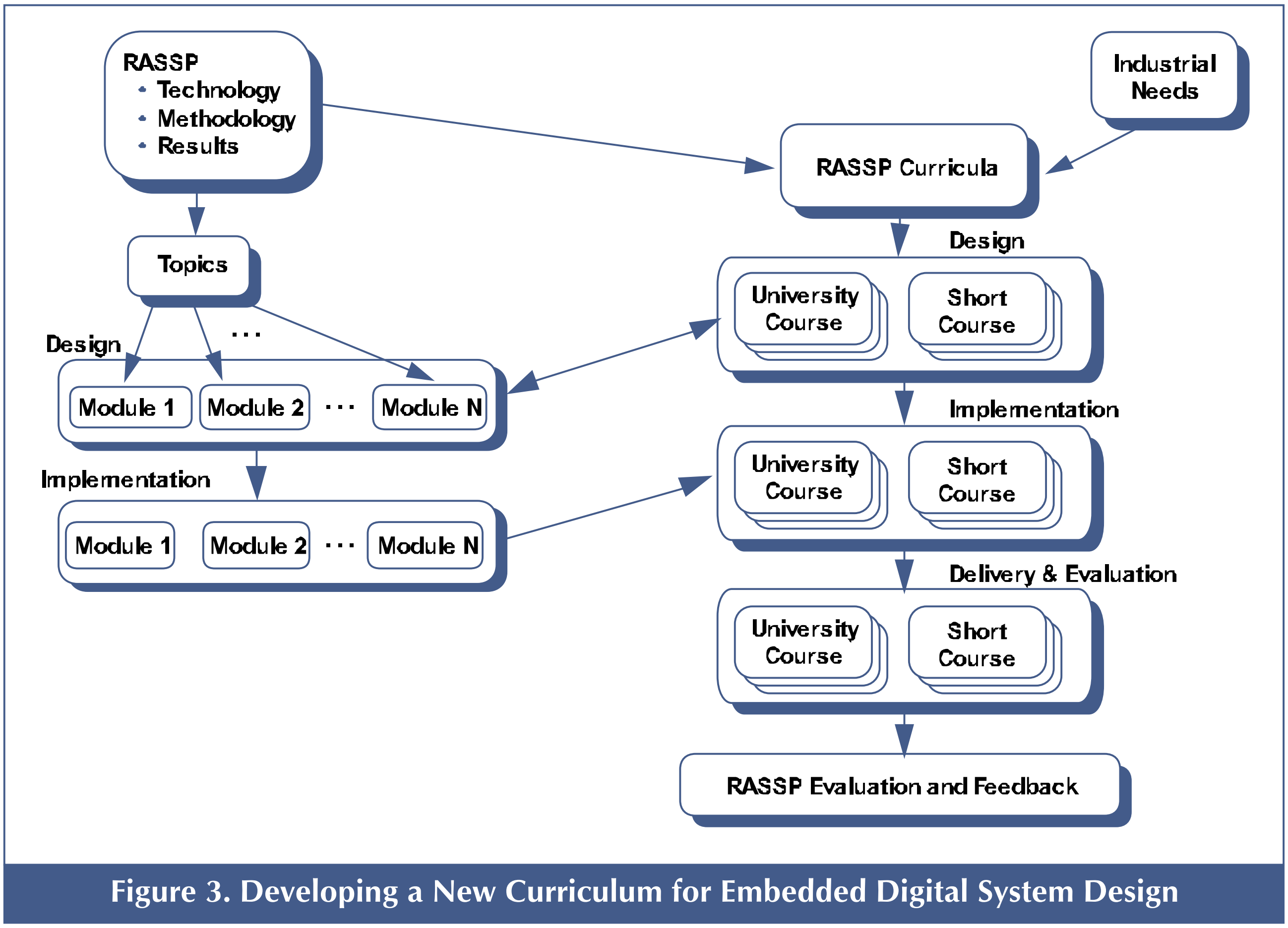


Figure 3. Developing a New Curriculum for Embedded Digital System Design

importance of top-down, language-based design techniques to meet the design challenges brought about by the rapid change in manufacturing capabilities, the results of this survey are significant. Indeed, this survey stimulated Robert Rozeboom, VP of Texas Instruments, to say, *"It's clear there is a significant need, and therefore, opportunity to increase the amount of training among undergraduate electrical engineering students at major universities worldwide* [4]*."*

Through the RASSP E&F program, DARPA and the Tri-Services are working to help universities adapt their curricula to incorporate RASSP technology so that tomorrow's graduates (BS, MS, Ph.D.) will be able to better meet the needs of industry and government. The approach being taken is to not only help the university community incorporate RASSP technology today, but also to establish a framework that will help the university community to effectively and efficiently respond to the ever changing needs of industry and government brought on by the dramatic rate of technological change.

One of the key challenges in the development of module-based RASSP curricula is determining what modules should be developed. The relationship between the module definition process and a module-based RASSP curriculum is presented in Figure 3. Using this technique, an example concentration area within a masters degree program is illustrated by the three courses shown in Table 1. These courses, RASSP 100, RASSP 101 and RASSP 102 have been piloted at the University of Virginia, the University of Cincinnati and the Georgia Institute of Technology. Currently more than 80 universities around the United States have obtained modules developed by the RASSP E&F team for use in their curricula.

The proposed university course sequence begins with RASSP 100, which focuses on rapid prototyping using VHDL, and explores the different levels of modeling, the relatively new practice of hardware/software codesign, and library reuse. It is followed by RASSP 101 and RASSP 102. RASSP 101 explores DSP algorithms and architectures in detail. RASSP 102 then examines some of the enterprise level issues associated with embedded digital system design. Not shown in Table 1, RASSP 103, is a project-based course designed to apply the principles learned from the earlier RASSP courses and to expose the student to a "near" industry scale problem. Table 1 also illustrates the flexibility of the modular approach.

By defining a module-based framework, proposing an example curriculum, and developing specific course material, the RASSP E&F program is creating an infrastructure that can help to ease transition from today's circuit design dominated curricula, to one that incorporates the spectrum of top-down design capabilities needed to design sophisticated, embedded digital systems. The most significant attribute of the module based approach is its ability to outlive the RASSP program and be ultimately "managed" by the education community itself. It is also generic in such a way that it can help universities more easily adapt to the rapidly changing state-of-the-art in embedded system design technology.

Table 1. Module Based RASSP Curriculum
(Shaded box indicates module is included in course)

Module	RASSP 100	RASSP 101	RASSP 102
System Level Modeling	▨	▨	
VHDL Basics	▨		
Structural VHDL	▨		
Behavioral VHDL	▨		
Advanced Constructs in VHDL	▨		
Hardware/Software Codesign	▨		
HW/SW Partitioning		▨	
DSP Architectures		▨	
Scheduling & Assignment for DSP		▨	
DSP Algorithm Design		▨	
Communication Protocols		▨	
RASSP Methodology Overview	▨	▨	
Requirements and Specification			▨
Virtual Prototyping for DSP Architectures		▨	
Virtual Prototyping using VHDL	▨		
Hardware Synthesis Overview	▨		
Libraries: Generation, Maintenance, and Reuse	▨		
Enterprise Integration			▨
Robust Design for Quality			▨
Project Management			▨
Test Technology Overview	▨		
Design for Manufacturing			▨
Cost Analysis for Design			▨
Performance Modeling using VHDL		▨	▨
Implementation Technologies		▨	

3.1 Transferring the E&F Education Framework

In addition to developing an infrastructure to assist in the transfer of the RASSP technology, the RASSP E&F team is leading a series of "teach the teacher" educator workshops. These workshops are designed to provide instructors with the understanding and material they need to include the latest technology in the classes they teach. The RASSP E&F team is working with other organizations, for example, the National Science Foundation and VHDL International (VI), to ensure that the widest possible benefits are derived from these workshops.

Working together with VI, several workshops have been held and others are scheduled. The focus of these workshops is on VHDL and its use in top-down design. (For more information see pages 6 & 9). VI is working with its members to ensure low or no cost access to VHDL software for universities. Together, the VI and RASSP E&F efforts are providing the impetus for addressing the need so clearly stated by Robert Rozeboom, VP of Texas Instruments. In addition, a Top-Down Design Workshop for Educators has been developed and is scheduled for mid-August at the University of Virginia.

4. Professional Education

In order to transfer RASSP technology to engineers, educators, and managers, the E&F Education team has devised a format in which the education modules can be delivered in three to five day short courses available to working professionals. Initially, a series of short courses modeled after the university courses RASSP100, 101, and 102 which included largely the same modules was developed. It is important to emphasize that the modules used in these short courses are exactly the same as those in university courses. Any required laboratory and other exercises accompanying the modules, however, required modifications to accommodate the more limited time constraints imposed by the short courses over the university courses.

More recently, however, the short course mechanism has been modified to address different audience's needs. The team has de-emphasized the predefined short course sequence and allowed industrial audiences to design their own short courses by selecting from the list of available modules in an almost *a la carte* fashion. The education team must, of course, ensure that any prerequisites are satisfied.

5. Executive Education

To help create a pull for the RASSP technology and professionals versed in that technology, a series of Executive Seminars have been developed and delivered. Targeting senior executives in industry and government, these seminars are designed to generate understanding and the desire to incorporate RASSP technology into their business by focusing on the business implications underlying the RASSP technology and presenting the business case behind this technology. To date, seminars have been given to senior managers in such places as Rockwell, Alliant Defense Electronics Systems, the National Security Agency (NSA), Texas Instruments, Allied Signal Corporation, NASA, and so on.

The seminars draw much of their material from the E&F educational modules. Unlike the university and short courses, however, seminars will only rarely include complete modules. Rather, they present the high-level concepts contained in the modules in order to communicate the advances of RASSP and encourage further inspection via additional courses or other RASSP transfer activities. The article by Jim Scharf, Sr., Mitchell Heller, and Larry Karns titled "Executive Education: Key to Implementing RASSP" explains executive education activities in more detail.

6. Conclusions

The RASSP Education and Facilitation team has been tasked with leading the RASSP technology transfer efforts. Very early in the process, the E&F team realized one of the key elements to a successful transfer of RASSP technology was the incorporation of the technology into academic programs. Realizing how hard it is to insert new technologies into a curriculum, the E&F team developed a novel approach to accomplish this task. This new approach, based on the development of technology modules, is currently being tested at the various universities of the team members. At the same time, the E&F team is initiating programs to see that this approach and the RASSP technology is adopted by the academic community. It is anticipated that the approach and the specific modules will be of significant benefit to the educational community, given the enormous task of keeping curricula current with the advances in microelectronics. An additional benefit of the module concept is the ability to reuse material for presentation in university courses, short courses, and executive seminars.

References

[1] A.B. Tucker and B. H. Barnes, “Flexible Design: A Summary of Computing Curricula 1991”, IEEE Computer, Vol. 24, No. 11, Nov. 1991, pp.56-66.

[2] ACM/IEEE-CS Joint Curriculum Task Force, *Computing Curricula 1991*, ACM Baltimore, MD., Order No. 201880, 1991.

[3] S.W. Director and R. A. Rohrer, “Reengineering the Curriculum: Design and Analysis of a New Undergraduate Electrical and Computer Engineering Degree at Carnegie Mellon University”, Proceedings of the IEEE, Vol. 83, No. 9, Sept. 1995, pp. 1246-1269.

[4] M. Jain, “Executive Director’s Message”, *VHDL Times*, Vol. 4, No. 3, page 2, Third Quarter 1995.

[5] *VHDL University Education Report*, from VHDL International, Santa Clara, CA.

Maximo Salinas, James Aylor, and Robert Klenke
Department of Electrical Engineering
University of Virginia
msalinas, jha, rhk2j @virginia.edu

Harold Carter
University of Cincinnati
ECI Dept., Mail Location 30
Cincinnati, OH 45221-0030
hal.carter@uc.edu

Vijay K. Madisetti
ECE,
Georgia Tech.
Atlanta, GA 30332-0250
vkm@ee.gatech.edu

Anthony J. Gadient
SCRA
5300 International Blvd.
N. Charleston, SC 29418
gadient@scra.org

Executive Education: Key to Implementing RASSP

Jim Scharf, Sr. and Larry Karns

Abstract

Regular readers of this Digest *are familiar with the RASSP goals, the technical challenges, and how they are being met. Of equal or greater importance to the overall success of the RASSP program is the widespread adoption of the tools and methods, architectures, and infrastructure by American industry. This can only be achieved when key executives are informed of the cost/benefit issues and convinced to make the investments necessary to adopt the RASSP technology. This article describes the RASSP Education and Facilitation (E&F) team efforts to document and present the business benefits of embracing RASSP achievements.*

1. Introduction

The RASSP program is now in its fourth year and has a significant list of achievements to its credit. These results can be accessed on the RASSP website at ***http://scra.rassp.org*** - one of the three highest rated websites in DSP.

RASSP is a program which was intended to benefit a wide segment of the American Electronics Industry. To ensure that the advancements developed under RASSP find their way into industry and academia, DARPA has funded a separate major task under RASSP for the purpose of Education and Facilitation to focus on:

- University Education
- Executive Education
- Single source of information.

The University Education component concentrates on creating a ***technology push*** by educating future engineers knowledgeable in the RASSP methodologies and architectures. This is accomplished through development and dissemination of university level course material for instruction in the technology and methodologies of RASSP. This has led to the acceptance of courses of instruction at a number of key universities which will ultimately result in a continuing supply of engineers skilled in RASSP methods.

The Information activity of the RASSP E&F program has the goal of disseminating information about the RASSP program and related activities. Mechanisms such as this *Digest* and the RASSP web site are used to accomplish this. The RASSP web site provides access to RASSP program information and links to many simulation models for digital circuits.

The Executive Education task has the goal of creating the ***technology pull*** or demand for RASSP methodologies and engineers trained in them. The targets for this effort are threefold. First, it is industry which creates jobs and builds products, and the adoption of the RASSP methodologies and approaches will increase the demand for the supply of skilled engineers. Second, purchasers of these needed services and systems, namely the government project and program offices, will help create demand if they request systems designed in the RASSP manner. Finally, demand can be created at the specific defense management colleges by training government program managers in the advantages of acquiring systems based on RASSP. This article focuses on the Executive Education activity of the RASSP E&F Program.

2. The Need for Executive Education

It is necessary to educate key executives and decision makers about the benefits that they can derive from RASSP adoption. To successfully educate any audience, the needs of the customer must be addressed in order to get and hold his/her attention and maintain interest while conveying information.

One goal of RASSP E&F is to communicate with executives who manage people that design digital signal processors and systems or who manage the procurement of those systems. They need to be informed of the benefits that they and their companies or agencies can derive from RASSP. In order to have the maximum impact on executives, a familiar venue with which they can relate needs to be utilized.

3. The Executive Education Approach

Educating the DSP community is necessary in order to create the demand for RASSP. Presentations or seminars are given to key individuals in the target organization. The RASSP E&F team works hand in hand with the RASSP prime contractors to ensure maximum success of these seminars. The components of the Executive Seminar are:

- Management Presentation including the business case for adopting the RASSP methodologies, architectures and tools (infrastructure),
- Top level technical presentations, and
- Technical demonstration(s).

The Management Presentation, including a RASSP business case has been prepared and delivered by the RASSP E&F team. The technical presentations and demonstrations are delivered by the RASSP prime contractors.

The demographics and desires of the target audience define the structure of the executive seminar (i.e., agenda for the day). The day can be the Management Presentation alone. This will be done by a member of the RASSP E&F team and can be accomplished with or without the attendance of one or more individuals from the RASSP prime contractors. Our experience has shown tremendous advantages when a member of the prime contractor's team is present to answer any of the more technical questions. This presentation, with questions, lasts about two hours and is aimed at the high level decision makers of an organization.

The second option is to add technical presentations, presented by a member of the RASSP prime contractors, to the management, or business case, presentation. This will add another couple of hours to the Management Presentation, so this Executive Seminar would last a half day. While the first part of the presentation is aimed at the high level decision makers in the organization, the target audience for the technical presentation is the technical manager and lead engineer.

A company, or government program office, can elect to have a day of RASSP by also adding technical demonstrations for the afternoon. The audience for these demonstrations would be similar to that for the technical presentations. If this option is chosen, there needs to be a great deal of discussion among the target audience, the RASSP prime contractors and the RASSP E&F team representative. There are many tools and parts of the methodology that can be demonstrated during this half day. An entire demonstration of everything involved in the RASSP program could take a week or more, so it is imperative that the major interest of the target audience be understood to maximize the benefit of the time spent during these demonstrations.

4. The Management Presentation

The purpose of the Management Presentation is to present the business case for the adoption of the RASSP methodologies, architectures and infrastructure. The presentation is organized into the following sections:

- Problem statement
- Efforts already being done outside of RASSP
- The need for RASSP
- The RASSP technical approach
- Results to date
- Business case analysis
- Next Steps.

The primary purpose of this presentation is to present a framework by which an organization, together with a RASSP prime contractor and RASSP E&F, can begin to calculate the organization specific benefits of adopting the RASSP technology. It is this presentation, along with the scheduling (in conjunction with the Government) of the executive seminars, that is the major focus of the Executive Education function of the RASSP E&F program.

4.1 Scheduling a Seminar

The process of scheduling and setting up an executive education seminar begins with suggestions of candidate organizations and initial contacts and telephone inquiries to validate the choice of organization, obtaining Government approval, and locating the proper facilitator at the candidate site. Any organization can become a candidate for an executive seminar by contacting the RASSP E&F program representative, Anthony Gadient, (803) 760-3376. Candidate organizations need to be involved in the development of digital signal processing, either for military or commercial markets, or they should be government representatives involved in the procurement of DSP systems. The facilitator at the site will be responsible for arranging a meeting with high level executives at his organization.

4.2 Prior to the Seminar

Prior to formalizing the agenda and schedule, the prospect is encouraged to visit the RASSP www site (***http://rassp.scra.org***) and to view a 15 minute video tape overview of the RASSP program produced by one of the prime contractors.

If only the single executive briefing can be achieved in the first meeting, all efforts are made to encourage follow up at a later date at the customer's site or at the prime vendor's site for the full technical lecture and demonstration(s).

4.3 Seminars Delivered to Date

To date the Management Presentation has been delivered at eight major organizations, including Texas Instruments, GEC Marconi, and Rockwell/Collins. Additionally, the Management Presentation has been delivered at government installations, such as Tinker Air Logistics Center.

4.4 Feedback

Subsequent to the delivery of an Executive Seminar, the attendees are asked to fill out a simple one page, two sided evaluation form regarding the achievement of objectives and to solicit ideas on how the experience could be improved for the attendee. Responses received to date have been positive, and a number of good suggestions have been offered which will be incorporated into future presentations.

One area that is being improved is the business case for RASSP. Executives are attentive to messages transmitted in terms of profits and losses, assets and depreciation, expenditures and savings and overall return on investment (ROI). While experiential data has been used to support the business case, it is very hard to come by good, independently verified business case data. The RASSP E&F Executive Education team is working diligently in seeking out good data and any feedback or suggestions would be welcomed.

The RASSP program itself is producing a significant volume of verified cost data. One of the key elements of the RASSP program is the development and measurement of specific performance improvements through benchmarks. These benchmark projects are defined and monitored by MIT Lincoln Lab, an entity independent of any of the RASSP prime contractors or tool vendors.

5. Back to Business

The current presentation already contains convincing evidence of the business reasons to adopt the benefits derived from RASSP. A basic business example from the current presentation is shown in Figure 1. This graph shows that in a commercial setting, a delay in bringing a product to market within a proper "market window" of time can cost a company in lost revenues and consequently lost profits. The case is made that RASSP accelerates the development process significantly, thus averting the potential loss.

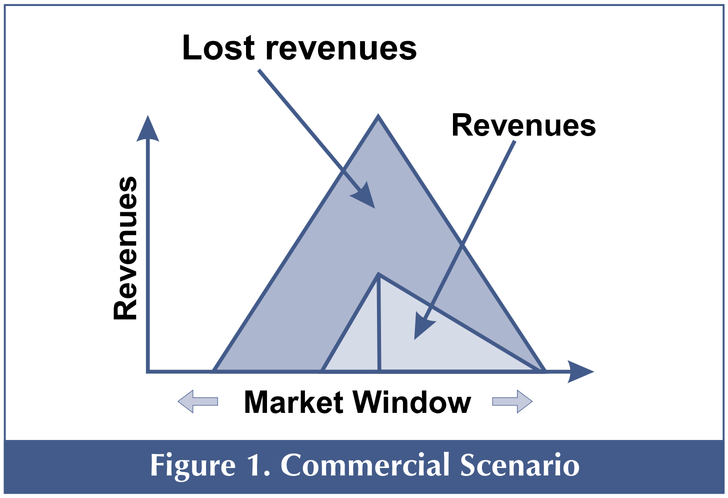


Figure 1. Commercial Scenario

DARPA's thrust with RASSP can have even more impact for military prime contractors. As shown in Figure 2, the typical scenario for systems development is centered around achieving a significant level of performance beyond current capabilities. This typically involves trying to push technology too far in the early years of a program with a point solution in order to achieve (possibly overly ambitious) goals, only to see system performance ultimately fall behind the commercial development curve in later years. The cost of the single point push at the inception of the project is high and artificially inflates systems cost, while the loss of capacity in the later years drives the early need for a replacement system - again at higher cost.

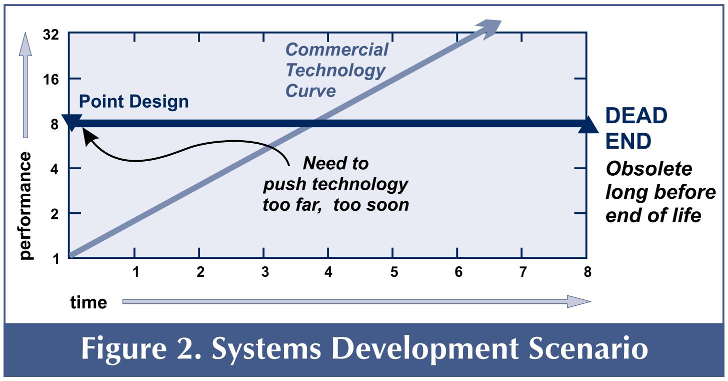


Figure 2. Systems Development Scenario

By contrast, RASSP's Model Year Architecture approach alleviates the problems on both ends of the time line, saving a great deal of cost. With the RASSP Model Year Architecture approach (displayed in Figure 3), a scalable, flexible approach is taken in the initial design phase, and all of the information is captured in a database. As time progresses, lessons are learned from the initial design, and the technology curve continues to advance. With a high degree of reuse guaranteed, it is far more cost effective to roll out improved designs in succeeding model years than the traditional "push hard, then age" scenario.

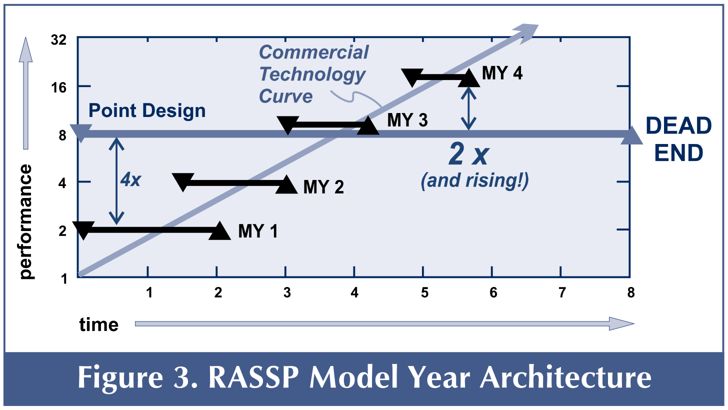


Figure 3. RASSP Model Year Architecture

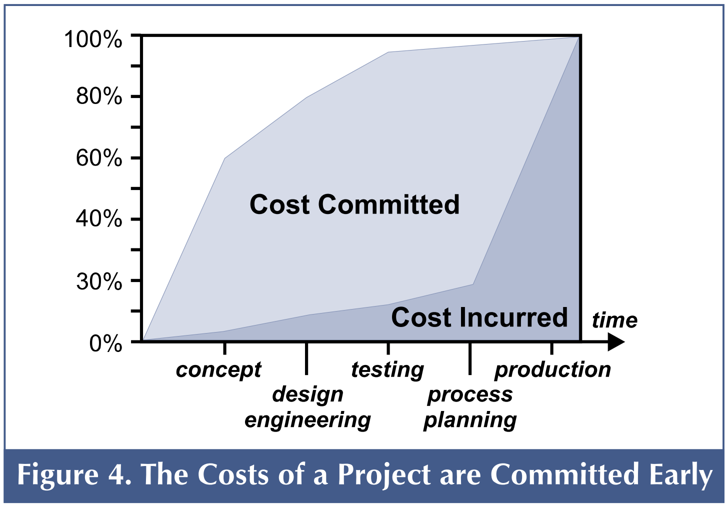


Figure 4. The Costs of a Project are Committed Early

Figure 4 shows how the costs of a project are committed early, long before a physical prototype can be produced. Once these costs are committed, it becomes too late to run "what if" studies or to explore alternative architectures and methods which might have improved the system performance or reduced costs or both.

Through the process of Virtual Prototyping, hierarchical models of trial systems can be built, explored and evaluated in a software virtual prototype. Actual data showing the benefits which can be achieved by the use of Virtual Prototyping, when presented as in Figure 5, makes a powerful case that is easily understood by engineer and executive alike. Direct enumeration of the actual average times required on a RASSP based system, which has been reduced from months to days, conveys a strong and unmistakable message. Graphics, such as these, coupled with compelling stories of actual use of RASSP techniques, help to make a forceful case for the executives who make the business decisions.

▷ **Board Fabrication**	**Avg. = 5 days**
▷ **Board Assembly**	**Avg. = 3 days**
▷ **HW Checkout**	**Avg. = 3 days**
▷ **HW/SW Driver Integ.**	**Avg. = 12 days**

Approx. 23 days
for Initial integration
(down from 18 months!)

Figure 5. Here is Why You Want Virtual Prototyping!

6. Continuing Improvement

Work is continuing with the RASSP prime vendors and MIT Lincoln Lab to develop additional business cases and amplify and focus the messages from the cases already in use. Collaboration is being actively pursued with other companies who have case history data for work that they have done using methods that are subsets of RASSP. Additional information is being sought from companies who wish to adopt some or all of RASSP's methodology, infrastructure and architecture and who are willing to participate in "before and after" case studies for future publication.

One significant piece of additional work that has recently begun is the development of a formal business case example for the adoption of RASSP by a DSP systems vendor. The initial case chosen for study was one developed by Lockheed Martin ATL under their contract to study cost models. They have produced a detailed comparison of the development of a new AWACS-like system with significant DSP requirements both under the RASSP methodology and in the conventional manner. This report provided the basic facts required to build a business case.

The model for building the actual case came primarily from the Department of the Army, "Economic Analysis Manual", U.S. Army Cost and Economic Analysis Center, July 1995. The results of the analysis showed that it was possible to utilize the cost savings from switching to the RASSP approach on the demonstration-validation phase alone to pay for all required tooling and training required by the adoption of RASSP and still produce an acceptable return on the initial investment (Table 1). In this case, the traditional methods would cost $12.9 Million for the Demonstration/Validation phase. Adopting RASSP techniques would require an initial investment of $2 Million, but would return 103% on the investment in two years.

Status Quo vs. RASSP

Phase	**Year**	**Savings**
Demonstration/Validation	0	$ (2,050,000)
Demonstration/Validation	1	$ 2,221,700
Demonstration/Validation	2	$ 3,908,300
Total Savings		**$ 4,080,000**
Net Present Value		$ 3,049,334
Internal Rate of Return		103%

Table 1.

Updates to the Executive Seminar presentation will be made as new information evolves. Likewise, the RASSP E&F Program will continue to make improvements to better achieve the goal of promulgating and proliferating the benefits of the RASSP program to American industry and government.

Jim Scharf, Sr.
Raytheon Electronic Systems
Software Engineering Laboratory
M/S-T3MJ26
50 Apple Hill Drive
Tewkesbury, MA 01876

Larry Karns
Arthur D. Little
5300 International Blvd.
North Charleston, SC 29418
karns@scra.org

MSE '97

Doing More with Less in a Rapidly Changing Environment

General Chair:
Don Bouldin, University of Tennessee

Program Co-Chairs:
Paul Hulina, Pennsylvania State University
James Aylor, University of Virginia

Publicity Chair:
Anthony Gadient, SCRA

Tutorial Chair:
Hal Carter, University of Cincinnati

This conference is cosponsored by the IEEE Computer Society Technical Committees on Design Automation, Test Technology and VLSI.

Partial support is being provided by the National Science Foundation and the Semi Conductor Research Corporation.

The meeting is also being held in cooperation with the ACM and ASEE.

IEEE International Conference on

MICROELECTRONIC SYSTEMS EDUCATION

July 21-23, 1997
Arlington, VA

This conference is dedicated to furthering undergraduate and graduate education in designing and building innovative microelectronic systems.

The meeting will focus on the following topics:

- Needs and Expectations of Industry
- Capabilities and Constraints of Academia
- Mixing Educational Concepts and Technology-Dependent Issues
- Infrastructure Support Provided by Industry and Government
- Mentoring and Partnering Among Institutions
- Using Complementary Research to Facilitate Education
- Distributed Learning What Works and What Doesn't
- Contributing and Sharing Educational Modules
- Project-oriented Education

The first day of the conference will be devoted to invited presentations and forums. The second day will consist of poster sessions for contributed papers and computer demonstrations and exhibits by CAD software and textbook vendors. The third day will feature tutorials that are included in the registration fee.

Keynote Speakers:

- Fred A. Augusti, Vice President of Messaging Systems, Lockheed Martin
- Edward A. Parrish, President, Worcester Polytechnic Institute
- William A. Wulf, President, National Academy of Engineering (tentative)

You can now register for MSE97 using a credit card via a secure link at: **http://secure.computer.org/conf/mse/register.htm**

Advance Registration Fees (Before July 7, 1997):

IEEE CS Members	$340
Nonmembers	$425
Full-time Students	$140

For more information, please contact the IEEE Computer Society, Volunteer Service Department at 202-371-1013 or s.wagner@computer.org.

The meeting will be held it the Crystal Gateway Marriott, 1700 Jefferson Davis Highway, Arlington, VA 22202. You should make your own hotel reservations no later than June 20th by calling 800-882-1043 or 703-920-3230. Be sure to mention you are with the MSE-97 meeting to obtain the group rate. You should plan to fly into National Airport.

Complimentary shuttle service is available to the hotel by calling from the baggage claim area. The hotel is connected to the Crystal City Metro stop.

RASSP Informational Activities

Jack Stinson, Tommy Taylor, Tom Egolf and Shahram Famorzadeh

Abstract

The purpose of the RASSP E&F Information Activity is to develop awareness and interest in RASSP technology by providing easy access to useful and well organized RASSP-related information. This article provides insight into two of the mechanisms that were implemented in order to provide access to the RASSP-related information: a newsletter and world wide web site. Additionally, the article looks at the upper hierarchy in which data is stored on the RASSP world wide web server.

1. Introduction

To develop awareness and interest in RASSP technology, the RASSP E&F program is providing universal access to RASSP information in a number of ways. A World Wide Web (WWW) site "***http://rassp.scra.org/***" for the RASSP program has been established. This WWW site was identified in *IEEE Spectrum* as one of the top-three web sites for Digital Signal Processing (DSP)[1]. The RASSP WWW site contains both general and specific RASSP information including a RASSP overview, technical documents from RASSP contractors, and educational material to support the teaching of the advanced methods being developed by the RASSP program. In addition to the RASSP WWW site, this periodical, entitled *The RASSP Digest*, is published that provides insight into the accomplishments and progress obtained in RASSP and other related programs. Technical articles, conference tutorials and papers are also developed to facilitate information distribution.

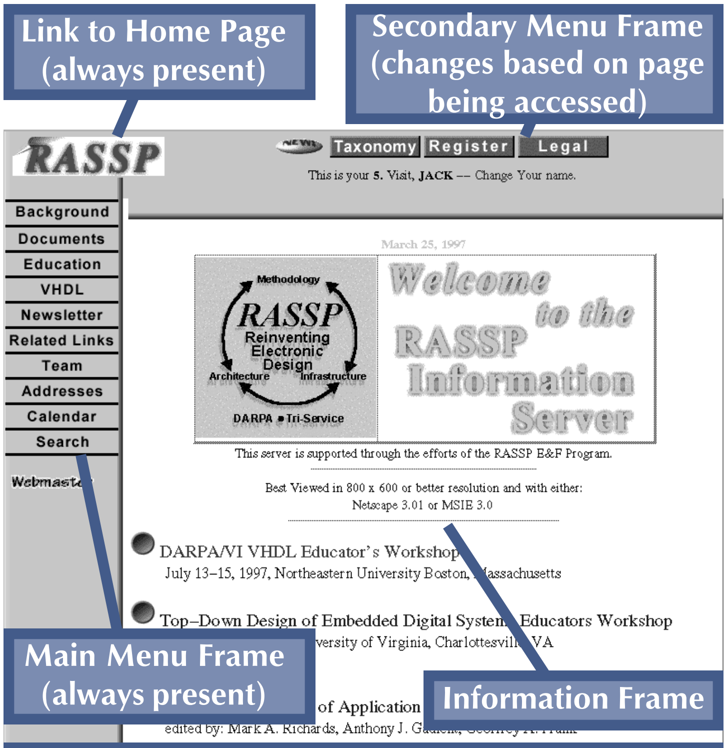
2. The RASSP Digest

The *RASSP Digest* provides periodic highlights of the RASSP program. Each issue has a focus and contains a wealth of information on the RASSP efforts in the focus area as well as articles from well-known authors outside the program. The current and all past issues are on the web server in multiple formats. The newsletter page may be accessed from the main menu. The list of topics covered to date are as follows:

- RASSP Education Activities – Vol. 4, June, 1997
- Technology Base Efforts – Vol. 3, September, 1996
- The Road to Enterprise Integration — Vol. 3, 1st. Qtr. 1996
- Model Year Architecture — Vol. 2, 4th. Qtr. 1995
- RASSP at 24 Months — Vol. 2, 3rd. Qtr. 1995
- 4X - Charting the Course — Vol. 2, 2nd. Qtr. 1995
- Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL) — Vol. 2, 1st Qtr. 1995
- RASSP After One Year — Vol. 1, 4th Qtr. 1994

3. RASSP Information Server

The RASSP Information Server contains information regarding all parts of the DARPA sponsored RASSP program. The format for the frames used on the RASSP server is seen in Figure 1. The "Main Menu Frame" on the left side is always present. The large "RASSP" in the upper left corner will return you to the homepage. The "Secondary Menu Frame" at the top of the page changes based on the page selected. Finally, the "Information Frame" in the middle of the page is where the information appears.



This is your 5. Visit, JACK –– Change Your name.

March 25, 1997

This server is supported through the efforts of the RASSP E&F Program.

Best Viewed in 800 x 600 or better resolution and with either:
Netscape 3.01 or MSIE 3.0

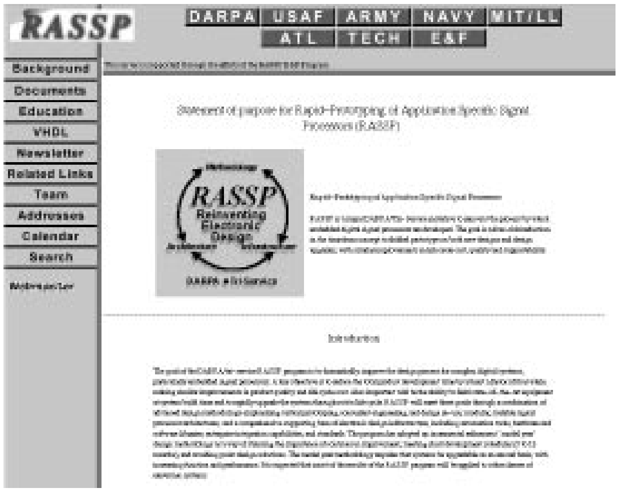
DARPA/VI VHDL Educator's Workshop
July 13–15, 1997, Northeastern University Boston, Massachusetts

Top–Down Design of Embedded Digital System Educators Workshop
versity of Virginia, Charlottesville VA

of Application
edited by: Mark A. Richards, Anthony J. G

Figure 1

The "Main Menu" buttons provides links to the major areas of focus for the RASSP server. This information includes background information about the program and program participants; documents; educational material and activities; VHDL models, programming guides and software; newsletter issues and sign-up; links to DSP and VHDL related sites; biographical information on the RASSP E&F team members; addresses of RASSP team members; calendar of RASSP and related events and a tool for searching documents on the RASSP server. To help you in your understanding the layout and contents of the RASSP server, selected areas will be described in the remainder of this article.



[illegible]

Statement of purpose for Rapid-Prototyping of Application Specific Signal Processors (RASSP)

Rapid-Prototyping of Application Specific Signal Processors

[illegible]

Introduction

[illegible]

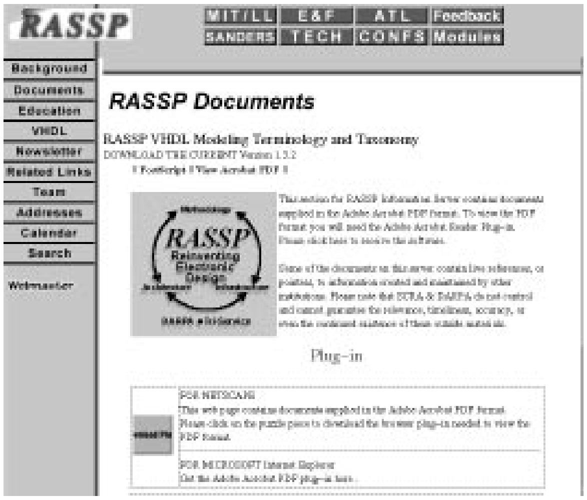
Figure 2

3.1 Background Information

The “Background Page”, seen in Figure 2, provides and overview of the RASSP Program and the Secondary Menu provides links to the government sponsors, MIT/LL Benchmarking program, Lockheed Martin ATL prime program, Technology Base programs and the Education and Facilitation program. The Technology Base page provides links to the RASSP pages of the 21 university and research institution efforts as well as an overview of these efforts in developing the support technology for the RASSP Program.

3.2 Documents

The “Documents Page”, as shown in Figure 3, is reached from the main menu. The Secondary Menu contains links to documents in the major areas of the program. The Information Frame contains papers or information that is highlighted and instructions on obtaining plug-ins for your browser to view some of the documents. The documents are mostly in HTML, Adobe Arcrobat (PDF) or PostScript. Some documents may also be in native format.



RASSP Documents

RASSP VHDL Modeling Terminology and Taxonomy
DOWNLOAD THE CURRENT Version 1.3.2
[PostScript | View Acrobat PDF]

This section for RASSP Information Server contains documents supplied in the Adobe Acrobat PDF format. To view the PDF format you will need the Adobe Acrobat Reader Plug-in. Please click here to receive the software.

Some of the documents on this server contain live references, or pointers, to information created and maintained by other institutions. Please note that SCRA & DARPA do not control and cannot guarantee the relevance, timeliness, accuracy, or even the continued existence of these outside materials.

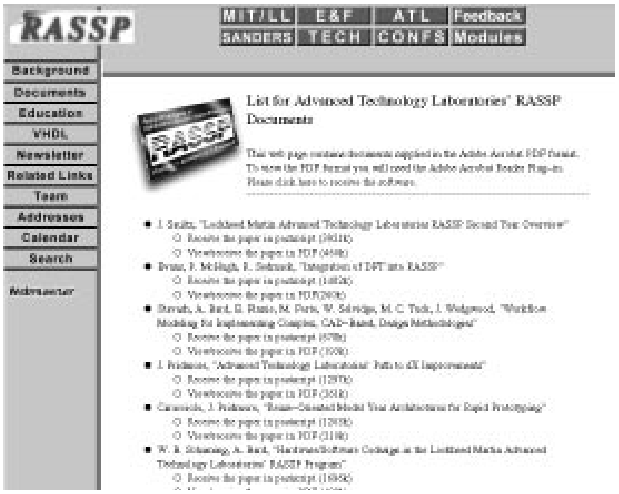
Plug-in

FOR NETSCAPE
This web page contains documents supplied in the Adobe Acrobat PDF format. Please click on the puzzle piece to download the browser plug-in needed to view the PDF format.

FOR MICROSOFT Internet Explorer
Get the Adobe Acrobat PDF plug-in here.

Figure 3

Figure 4 is the Lockheed Martin Advanced Technology Laboratories (ATL) documents page and shows an example of the format that is used.



List for Advanced Technology Laboratories' RASSP Documents

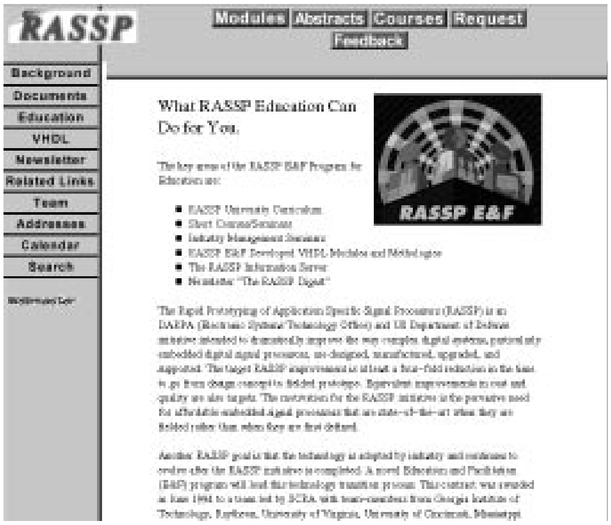
This web page contains documents supplied in the Adobe Acrobat PDF format. To view the PDF format you will need the Adobe Acrobat Reader Plug-in. Please click here to receive the software.

- J. Saultz, "Lockheed Martin Advanced Technology Laboratories RASSP Second Year Overview"
  - Receive the paper in postscript (3901k)
  - View/receive the paper in PDF (450k)
- Evans, P. McHugh, R. Sedmak, "Integration of DFT into RASSP"
  - Receive the paper in postscript (1482k)
  - View/receive the paper in PDF(290k)
- Hurtado, A. Bard, E. Rundo, M. Parris, W. Selridge, M. C. Tsch, J. Wedgwood, "Workflow Modeling for Implementing Complex, CAD-Based, Design Methodologies"
  - Receive the paper in postscript (478k)
  - View/receive the paper in PDF (100k)
- J. Pridmore, "Advanced Technology Laboratories' Path to 4X Improvement"
  - Receive the paper in postscript (1297k)
  - View/receive the paper in PDF (361k)
- Carmichael, J. Pridmore, "Reuse-Oriented Model Year Architectures for Rapid Prototyping"
  - Receive the paper in postscript (1293k)
  - View/receive the paper in PDF (318k)
- W. B. Schaming, A. Bard, "Hardware/Software Codesign in the Lockheed Martin Advanced Technology Laboratories' RASSP Program"
  - Receive the paper in postscript (1595k)

Figure 4

3.3 RASSP Education And Facilitation Information

The “Education Page”, provides access to information and material created by the RASSP E&F effort. Figure 5 shows some of the items that are available. One of the major items that the RASSP E&F program has accomplished is the creation of university level course modules that cover a variety of important topics. These are available free of charge electronically to qualified professors. The “Modules” button provides a list of the modules and a link to a request form. The “Abstracts” button takes you to a link that contains a full page description of each module. The “Request” button will take you to a form to request the course modules and the “Feedback” button allows users to provide feedback on the modules by completing a form. The “Courses” button provides information on RASSP E&F courses and workshops.



What RASSP Education Can Do for You.

The key areas of the RASSP E&F Program for Education are:

- RASSP University Curriculum
- Short Courses/Seminars
- Industry Management Seminars
- RASSP E&F Developed VHDL Modules and Methodologies
- The RASSP Information Server
- Newsletter "The RASSP Digest"

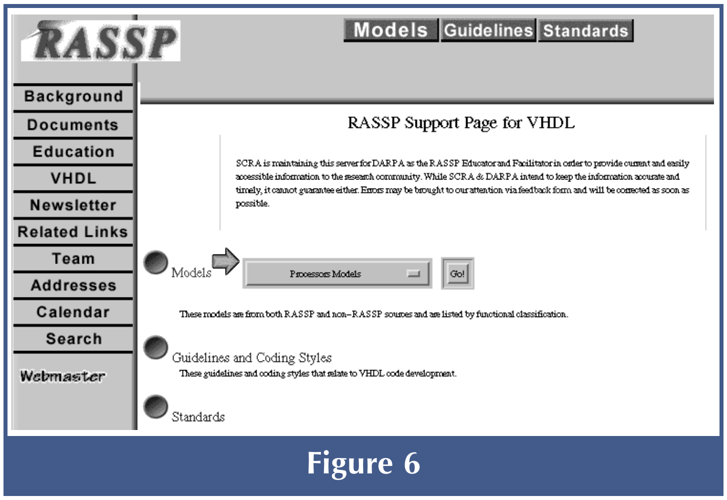
The Rapid Prototyping of Application Specific Signal Processors (RASSP) is an DARPA (Electronic Systems Technology Office) and US Department of Defense initiative intended to dramatically improve the way complex digital systems, particularly embedded digital signal processors, are designed, manufactured, upgraded, and supported. The target RASSP improvement is at least a four-fold reduction in the time to go from design concept to fielded prototype. Equivalent improvements in cost and quality are also targets. The motivation for the RASSP initiative is the pervasive need for affordable embedded signal processors that are state-of-the-art when they are fielded rather than when they are first defined.

Another RASSP goal is that the technology is adopted by industry and continues to evolve after the RASSP initiative is completed. A novel Education and Facilitation (E&F) program will lead this technology transition process. This contract was awarded in June 1994 to a team led by SCRA with team-members from Georgia Institute of Technology, Raytheon, University of Virginia, University of Cincinnati, Mississippi

Figure 5

3.4 VHDL Models and Information

The “VHDL Page” is the focal point for VHDL Models and Information. Figure 6 shows some of the information that is available. Each of these pages is linked to a wealth of information in the selected topic area.



RASSP Support Page for VHDL

SCRA is maintaining this server for DARPA as the RASSP Educator and Facilitator in order to provide current and easily accessible information to the research community. While SCRA & DARPA intend to keep the information accurate and timely, it cannot guarantee either. Errors may be brought to our attention via feedback form and will be corrected as soon as possible.

Models

These models are from both RASSP and non-RASSP sources and are listed by functional classification.

Guidelines and Coding Styles

These guidelines and coding styles that relate to VHDL code development.

Standards

Figure 6

Figure 7 shows the representative tree structure of the topic areas and the population in each of these areas that can be accesses from the VHDL page.

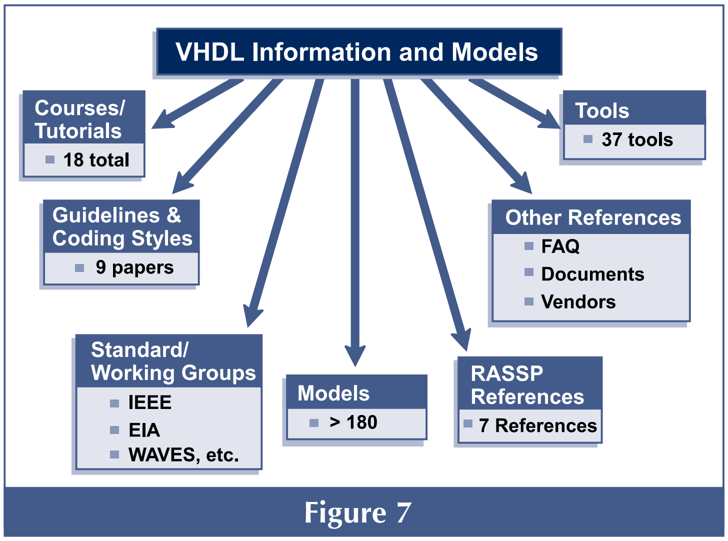


Figure 7

One of the major links under the “VHDL Page” is the “VHDL Models Page”. Figure 8 provides an overview of the different model functionalities available and some of the individual model types available in each function. Most of these models reside on the RASSP server, but some links are to other servers.

When you select the VHDL button on the Main Menu a disclaimer box will appear. Some of the box is shown in Figure 9. This box contains disclaimer and copyright information about the VHDL Models and material. This disclaimer must be read and accepted before you download any information.

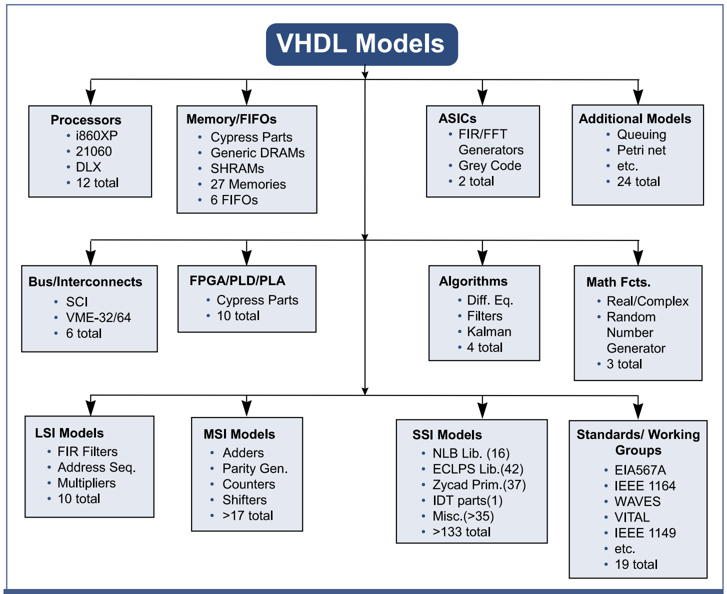


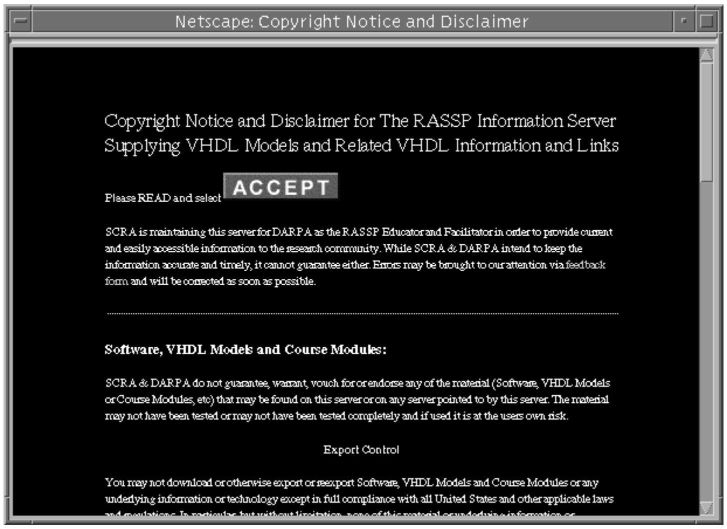
Figure 8

3.6 Search Tools

The RASSP Information Server “Search Tool” allows users to perform an integrated electronic full-text search with Boolean operations on the RASSP Information Server. The “Search Tool” page is shown in Figure 10. This search is available only on documents residing on the RASSP server. A search on “Harr and VHDL” will return all the documents that contain both “Harr” and “VHDL”. Note that the search is not case sensitive and “harr and vhdl” will return the same list of documents.

4. Conclusion

The RASSP E&F Information activities are aimed at providing single point access to RASSP Program information. This article has discussed two major mechanisms which help achieve this objective. Accesses to the RASSP Information Server has been



Copyright Notice and Disclaimer for The RASSP Information Server
Supplying VHDL Models and Related VHDL Information and Links

Please READ and select ACCEPT

SCRA is maintaining this server for DARPA as the RASSP Educator and Facilitator in order to provide current and easily accessible information to the research community. While SCRA & DARPA intend to keep the information accurate and timely, it cannot guarantee either. Errors may be brought to our attention via feedback form and will be corrected as soon as possible.

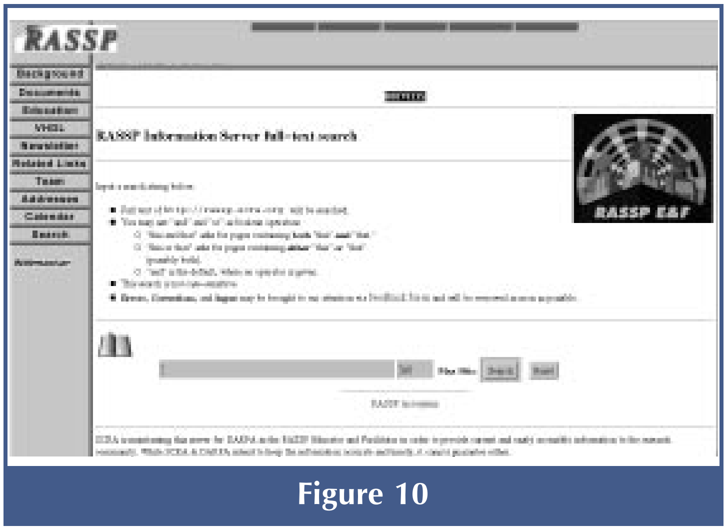
Software, VHDL Models and Course Modules:

SCRA & DARPA do not guarantee, warrant, vouch for or endorse any of the material (Software, VHDL Models or Course Modules, etc) that may be found on this server or on any server pointed to by this server. The material may not have been tested or may not have been tested completely and if used it is at the users own risk.

Export Control

You may not download or otherwise export or reexport Software, VHDL Models and Course Modules or any underlying information or technology except in full compliance with all United States and other applicable laws

Figure 9



RASSP

Background
Documents
Education
VHDL
Newsletter
Related Links
Team
Addresses
Calendar
Search

Webmaster

RASSP Information Server full-text search

[illegible]

- [illegible]
- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- [illegible]
- [illegible]

RASSP E&F

[illegible] Search [illegible]

[illegible]

[illegible]

Figure 10

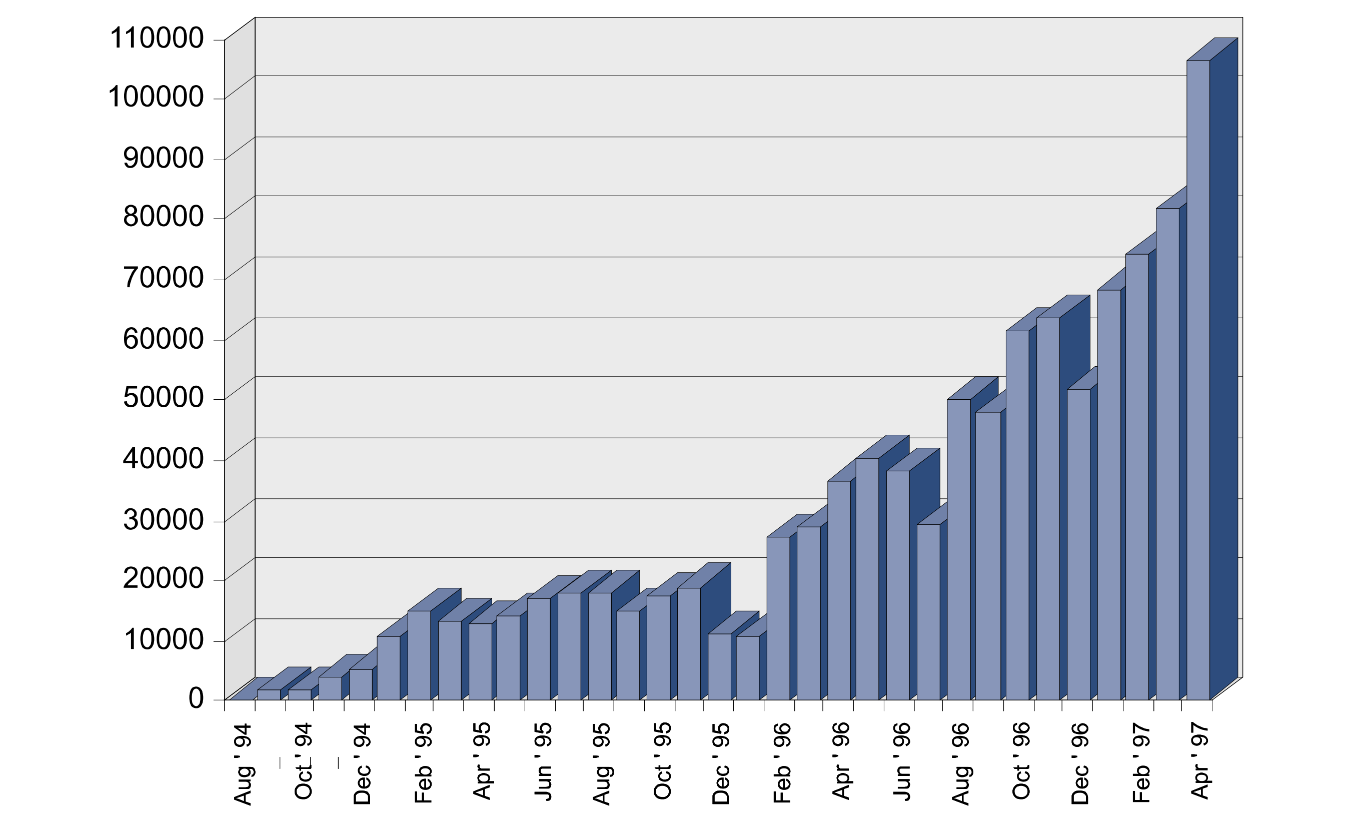
steadily increasing (see Figure 11) with over 1,000,000 since it began. To see for yourself the wealth of information available, visit us at ***http://rassp.scra.org/***. The contents of the RASSP server are updated on a regular basis and any feedback on the web server is appreciated. You can send feedback by selecting the "Webmaster" button at the bottom of most pages or by sending email to info@rassp.scra.org.

References

[1] R. Comerford, Editor, "Web Sights, Searching the Web", *IEEE Spectrum*, page 116, January 1996.

Jack Stinson, Tommy Taylor
SCRA
5300 International Boulevard
N. Charleston, SC 29418
stinson@scra.org**,** taylor@scra.org

Tom Egolf, Shahram Famorzadeh
ECE
Georgia Tech.
Atlanta, GA 30332-0250
egolf@ece.gatech.edu
shahram@eedsp.gatech.edu



Total accesses since the beginning of the program are over 1,000,000.

Figure 11

Calendar of Events

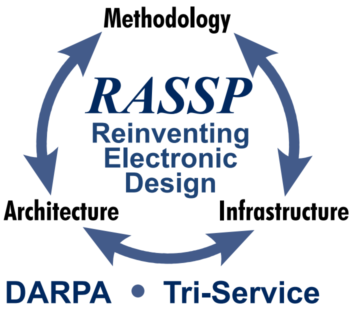
Design Automation Conference (DAC)	June 9-13	Anaheim, CA	WWW.DAC.COM
Microelectronic Systems Education	July 21-23	Arlington, VA	www.cedcc.psu.edu/mse97
DARPA/VI VHDL Educator's Workshop	July 13-15	Northeastern University Boston, MA	navabi@ece.neu.edu
Top-Down Design of Embedded Digital Systems Educators Workshop	August 11-14	University of Virginia, Charlottesville, VA	johnson@scra.org
VHDL Performance Modeling Workshop	November 16-17	Charlottesville, VA	johnson@scra.org
VHDL International Users' Forum (VIUF)	Oct 19-22, 1997	Hyatt Regency Crystal City Hotel Washington, D.C.	www.vhdl.org

DARPA/VI VHDL Educator's Workshop

Northeastern University Boston, Massachusetts
July 13-15, 1997

A third DARPA/VI VHDL Educator's Workshop is being sponsored by Viewlogic and VHDL International (VI), due to the overwhelming response to the two DARPA/VI VHDL Educator's Workshops. This workshop, to be held in Boston, Massachusetts, will be hosted by Professor Zain Navabi of Northeastern University.

The workshop is open to academia and industry. The cost for qualified educators is $200 and for industry is $1000. This fee includes the course material by DARPA, 3-day lecture and laboratory work by Northeastern University, and Viewlogic's VHDL Simulation and Synthesis software PC license by Viewlogic.



This workshop is designed for educators interested in introducing VHDL into their undergraduate and graduate curriculum, and designers or managers wanting to incorporate VHDL in their digital design flow. The workshop will be presented in four modules: Basic VHDL, Structural VHDL, Behavioral VHDL and Advanced Concepts in VHDL. Instructional material will be supplied that can be used in undergraduate and graduate courses, as well as staff training or reference. This material will include notes, labs, test problems and answers. In addition, through a grant provided by Viewlogic, a copy of the Viewlogic simulator used in the laboratory will be provided to each participant in the workshop. Information will also be provided on VI's education program. VI has made arrangements with member companies to provide free VHDL software to qualified educational institutions.

Lodging accommodations may be obtained from The Colonnade Hotel. The rate is $116/single and $141/double and reservations can be made by calling (617) 424-7018. Reservations must be made by June 28, 1997, to take advantage of the group rate. Refer to the "VHDL Educator's Workshop" when making the reservation.

More information is available by contacting Dr. Zain Navabi, Northeastern University, at navabi@ece.neu.edu or by visiting our web site at http://rassp.scra.org/VHDL.WORKSHOP.

For additional RASSP information including an electronic version of this publication, visit the RASSP World Wide Web site at

http://rassp.scra.org

Recognized by IEEE as one of the top three WWW sites on DSP.

IEEE Spectrum, January, 1996

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • ADL

RASSP Steering Committee

ARPA (ETO)		
	Randy Harr	**Program Manager**
ARMY		
	Randy Reitmeyer	**Administrative COTR, Lockheed Martin - Advanced Technology Laboratories**
	Arne Bard	**Technical COTR, Lockheed Martin - Advanced Technology Laboratories**
NAVY		
	Ingham Mack (ONR)	
	Gerry Borsuk (NRL)	
	J. P. Letellier (NRL)	**Administrative COTR, Lockheed Martin - Sanders**
		Technical COTR, Lockheed Martin - Sanders
AIR FORCE		
	Stan Wagner	**Educator Facilitator and Technology Base**
	John Hines	**COTRs**

RASSP Digest-Rapid Prototyping of Application Specific Signal Processors

The RASSP Digest is published quarterly and provides information for and about the RASSP Program and rapid systems development. For more information, contact Dr. Anthony Gadient or Dr. Vijay Madisetti, Editors, at the addresses below:

Anthony J. Gadient
Phone: 803-760-4082
FAX: 803-760-3349
Email: gadient@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

Vijay K. Madisetti
Phone: 404-853-9830
FAX: 404-853-9171
Email: vkm@ee.gatech.edu
Georgia Tech
School of Elec. & Computer Eng.
Atlanta, GA 30332-0250

Bryan R. Bryant
Managing Editor
Phone 803-760-3363
Email: bryant@scra.org
SCRA
5300 International Boulevard
North Charleston, SC 29418

RASSP E&F
SCRA • GT • UVA • Raytheon
UCinc • EIT • ADL

Up Coming Educator Workshops

Presented by **DARPA** and **RASSP E&F**

For more information contact
http://rassp.scra.org/ or 803-760-3376

☑ TOP-DOWN DESIGN WORKSHOP

August 11-14, 1997

For Educators interested in introducing Top-Down design using VHDL into their undergraduate and graduate curriculum

☑ VHDL Performance Modeling Workshop

November 16&17, 1997

This workshop will provide an introduction to the use of VHDL in system level performance modeling.

SCRA
5300 International Blvd.
N. Charleston, SC 29418

APPENDIX F - *Rapid Prototyping Of Application Specific Signal Processors*

Published By Kluwer Publishers (book)(external.)

Please search Kluwer Academic Publishers (www.wkap.nl) to view more information about this book.

Book Title:

Rapid Prototyping of Application Specific Signal Processors

edited by

Mark A. Richards
Georgia Tech Research Institute, Atlanta, USA

Anthony Gadient
South Carolina Research Authority, Charleston, USA

Geoffrey A. Frank
Research Triangle Institute, NC, USA

Kluwer Academic Publishers, Boston
ISBN 0-7923-9871-8
February 1997, 204 pp.